Licenciatura em Engenharia Informática

Modelo de Classificação e Predição de Preços de Áreas Imobiliárias da cidade de Lisboa

Relatório do Projeto de Final de Curso

Projeto Realizado Sob a Orientação de:

Prof. Doutor Nuno Brás

Ano Letivo: 2017/2018

David Bonifácio e Joana Martins
UNIVERSIDADE AUTONOMA DE LISBOA

ImoPrediction

David Novais Bonifácio e Joana Rita Monteiro Martins

Relatório submetido com requisito parcial para obtenção do grau de licenciatura em Engenharia Informática

Orientado por Prof. Doutor Nuno Brás na Universidade Autónoma de Lisboa

Universidade Autónoma de Lisboa

Julho, 2018

Agradecimentos

Este trabalho não ficaria completo sem agradecer a todos os que nos ajudaram a concretizá-lo. Em primeiro lugar, gostaríamos de dar uma palavra especial de agradecimento ao nosso orientador, Professor Doutor Nuno Brás, pelo apoio, disponibilidade e compreensão demonstrados ao longo do período de execução deste trabalho. Gostaríamos também de agradecer a todos os nossos amigos e familiares que de certa forma nos deram o seu apoio e alguma sugestão para o desenvolvimento deste trabalho.

A todos o nosso muito obrigado.

Resumo

Este documento é um relatório de um projeto que foi realizado no âmbito da cadeira de Laboratório de Projeto e tem por objetivo dar a conhecer o trabalho realizado e as diferentes fases do seu desenvolvimento. O título do projeto, Modelo de Classificação e Predição de Áreas Imobiliárias da cidade de Lisboa, dá alguma indicação do intuito deste que, de facto, consiste na previsão de valores de casas consoante múltiplas características que o utilizador coloque, sendo feito depois a previsão recorrendo a modelos de previsão que iremos salientar posteriormente.

O documento é constituído por 7 capítulos nos quais se apresenta o trabalho realizado no decorrer do semestre.

O capítulo II aborda, de forma geral, os objetivos do projeto, ou seja, apresenta as funcionalidades que este possui dando um contexto geral, caracterizando cada funcionalidade existente no projeto.

No capítulo III é feita a descrição detalhada do projeto, ou seja, neste capítulo poderá encontrar as tecnologias usadas durante o processo de desenvolvimento, descrições mais detalhadas do porquê e como implementado cada uma das mesmas. Poderá também ver esquemas que ilustrem tudo o que foi referido anteriormente. Em suma, este é o capítulo mais importante, pois é onde é explicado ao pormenor cada passo da realização do nosso projeto, desde as análises de requisitos, ao desenvolvimento completo do projeto.

No capítulo IV encontra-se o projeto desenvolvido, por outras palavras, neste capítulo fazemos, as demonstrações do que o nosso projeto faz, realizando assim pequenos testes para demonstração da solução desenvolvida e as configurações dos componentes constituintes do sistema.

No capítulo V realizamos algumas conclusões sobre o projeto, aspetos que podiam ser melhorados e aspetos que podiam estar implementados, de forma a que o projeto fique o mais completo e bem desenvolvido. Neste capítulo, fala-se também das dificuldades encontradas na realização do projeto, sendo neste capitulo onde serão apresentadas as conclusões finais.

No VI capítulo são apresentadas as referências utilizadas no decorrer do projeto.

Por fim, no capítulo VII, são apresentados os anexos.

Índice Geral

Índice Figuras

Glossário

API - Interface de programação de aplicações

Web Page – Designa a rede que conecta computadores por todo mundo, a World Wide Web (WWW).

Machine Learning - Aprendizagem de máquina, é um subcampo da ciência da computação que evoluiu do estudo de reconhecimento de padrões e da teoria do aprendizagem computacional em inteligência artificial.

Datawarehouse - Armazém de dados é utilizado para armazenar informações relativas às atividades de uma organização em bancos de dados, de forma consolidada.

Site - Web site, conjunto de web pages

URL - Localizador Uniforme de Recursos, refere ao endereço de rede no qual se encontra algum recurso informático

Framework - É uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.

Features - Palavra usada em Machine Learning para caraterísticas

SGBD - Sistema de gestão de base de dados

Bibliotecas - É uma coleção de subprogramas utilizados no desenvolvimento de software

Dataset - É uma coleção de dados normalmente em tabelas

Outliers - É uma observação que apresenta um grande afastamento das demais

Login - É o processo para aceder um sistema informático restrito feita através a autenticação

Capítulo 1

Introdução

Comprar ou alugar um imóvel nas grandes cidades está cada vez mais caro e a tendência é para que os preços continuem a agravar-se. Os preços das casas continuam a bater recordes e a tendência é para se manter. O risco de Portugal já estar ou a caminhar a passos largos para uma bolha imobiliária, pelo menos nas grandes cidades, já fez soares alarmes. Segundo a Comissão Europeia, o aumento dos preços das casas foi principalmente, restrito às áreas turísticas, já que, no geral, os valores no setor da habitação ainda estão a recuperar da queda da crise.

Com tudo o que se têm passado em volta do ramo imobiliário e como ainda nada do que implementamos existe no mercado, decidimos desenvolver um projeto que desse ao utilizador uma maneira de este conseguir prever o preço de uma casa tendo por base as características que o mesmo quer para comprar ou alugar.

Este projeto está desenvolvido por várias partes, desde a recolha de informações a sites imobiliários, à introdução destas informações numa base de dados, na realização de métodos de previsão, na criação de uma API para resposta sobre informações contidas na base de dados e para resposta aos métodos, e por fim à apresentação de todos estes mecanismos numa Web Page.

Assim sendo, ao longo deste relatório iremos explicar cada um dos campos de desenvolvimento de forma detalhada, para que se possa entender quais as tecnologias utilizadas, quais as formas de utilização e quais as formas de contornar os erros que nos apareciam. Quando chegarmos à parte das previsões, iremos fazer algumas ilustrações, recorrendo a gráficos para que se entenda, o quanto os preços das casas dispararam em certos sítios da cidade e como é que com algumas características os preços das casas podem alterar para valores exorbitantes.

Após explicação de cada módulo iremos fazer uma representação do que o nosso modelo é capaz de fazer, explicando também detalhadamente as suas funções.

De referir que este projeto requereu bastantes conhecimentos e domínio sobre algumas linguagens e ferramentas que nunca antes tínhamos utilizado. Tudo o que será apresentado foi desenvolvido por nós, através de muitas pesquisas e muito trabalho envolvido.
Em suma, de referir que este projeto é realmente desafiante e à medida que se for explicando em maior detalhe cada um dos módulos dá para entender o quanto o projeto é trabalhoso, mas ao mesmo tempo bastante interessante, devido à parte de machine learning que na nossa opinião dá todo um outro rumo a este projeto.

Capítulo 2

Objetivos e Contexto

2.1 Projeto ImoPrediction

O projeto ImoPrediction visa a disponibilidade de um site imobiliário em mostrar ao seu utente previsões de casas conforme as características que o mesmo deseja. Este projeto é um projeto revolucionário visto que, não existe nada parecido no mercado, uma vez que se junta a um site imobiliário uma vertente de Machine Learning, renovando totalmente o conceito.
Em termos operacionais, são objetivos do ImoPrediction:

- Determinar fontes credíveis de informação sobre preços e características de casas;
- Reunir a informação num datawarehouse;
- Ilustrar preços de casas reais, tendo uma secção própria para o efeito;
- Prever preços de casas consoante os parâmetros que o utilizador desejar, através de modelos de classificação;
- Criar uma API de acesso que permita, com base num conjunto de parâmetros, interrogar o modelo gerado;
- Renovar o conceito dos sites imobiliários, dando uso de tecnologias que nos dias de hoje têm começado a ter grande impacto.

Para atingir os objetivos do projeto, desenvolveu-se um modelo por onde nos guiamos para que tudo corre-se dentro do previsto. Maior parte do trabalho foi desenvolvido em Python, apoiando-se em tecnologias como Scrapy, MySQL, Scikit-Learn, Flask, Javascript, HTML e CSS.

Numa primeira fase, o ImoPrediction, não passava mais do que um simples site de imobiliária comum, como o site da imobiliária Century 21. Mas após implementação dos métodos de previsão, através do Scikit-Learn, passou a ser um site com novas funcionalidades e a cumprir o requisito do projeto que escolhemos.

Ao longo do relatório, poderá ver todas as etapas da realização do projeto e perceber melhor os conceitos e metodologias que utilizamos.

Capítulo 3

Metodologias

3.1. Metodologia

A utilização de uma metodologia de desenvolvimento, também denominado de processo de software, é um fator primordial para o sucesso das empresas de desenvolvimento de software. Um processo de software pode ser entendido como um conjunto estruturado de atividades para desenvolver um sistema de software.

A metodologia de desenvolvimento adotada, na execução das tarefas que envolveram o desenvolvimento deste projeto final, foi o seguinte modelo:

Figura 1. Modelo de desenvolvimento do projeto

A fase de análise é caracterizada pela identificação dos requisitos da aplicação, que consiste usualmente nos serviços que se devem fornecer, limitações e objetivos do software. Na fase de desenho são desenvolvidos modelos conceptuais e é analisada a melhor solução de implementação dos componentes da aplicação, tomando-se decisões fundamentais na conceção de novos componentes ou das alterações a fazer aos componentes já existentes. Na fase de implementação consiste na codificação da aplicação numa linguagem de programação. Na fase de testes são executados testes de forma a solucionar erros de implementação e assegurar que são produzidos resultados que coincidam com os requisitos especificados. Nesta fase é produzida toda a

documentação relativa à aplicação. Finalmente, a última fase consiste na integração da aplicação no sistema, ou seja, corresponde à entrega do produto.

De referir, que o modelo anterior modelo segue uma abordagem *top-down* e tem a vantagem de só avançar para a tarefa seguinte depois de nos certificarmos que a anterior se encontra em condições. Contudo, apresenta como desvantagem a dificuldade em responder a mudanças nos requisitos depois do processo se ter iniciado.

Capítulo 4

Trabalho Desenvolvido

Este capítulo detalha todo o trabalho realizado por nós ao longo do semestre e está organizado em seis subcapítulos, sendo estes compostos, um pelo modelo geral do projeto e os outros cinco referentes às subpartes do projeto. Cada um destes cinco subcapítulos, iram conter a análise de requisitos, as tecnologias utilizadas, e a explicação detalhada de como aplicamos cada uma.

4.1 Modelo do Projeto

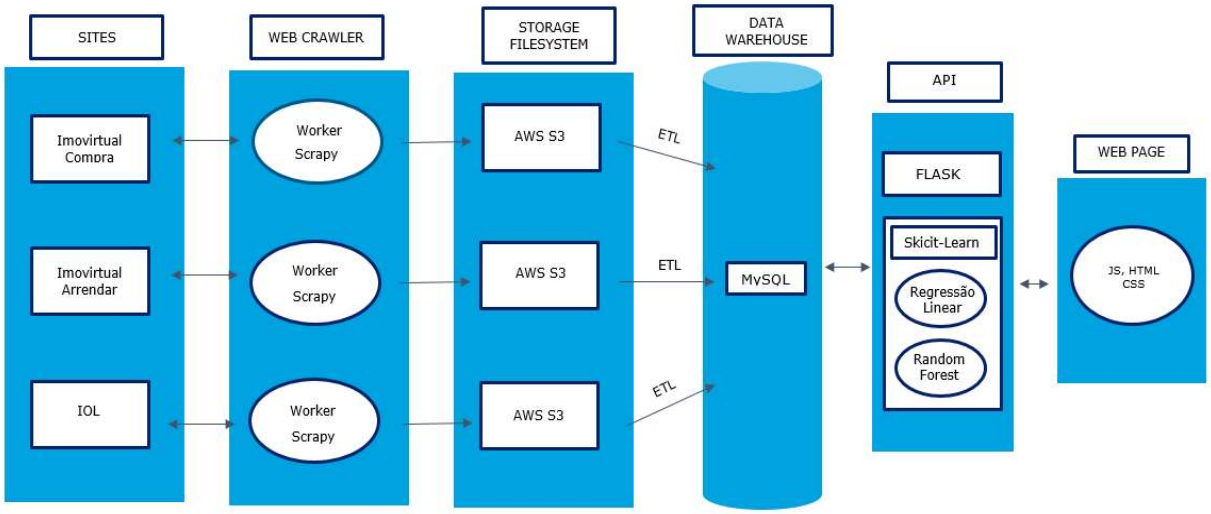


Figura 2. Modelo de desenvolvimento do Projeto

Ao analisarmos o enunciado do projeto, deparamo-nos com o facto de termos de planear como iriamos desenvolver o projeto. Posto isto, começamos a dividir o trabalho por secções e realizamos o seguinte modelo que está representado na Figura 2. Este modelo ilustra todos os parâmetros desenvolvidos por nós para que o projeto funcione como previsto.

A análise do modelo irá ser realizada por partes, para que seja mais fácil a leitura e a compreensão do que realizamos. Iremos dividir o modelo em cinco partes essenciais, sendo elas: os Web Crawlers, o Data Warehouse, os Modelos, a API e a Web Page.

4.1.1 Web Crawlers

4.1.1.1 Análise de requisitos

Na parte inicial do projeto, necessitávamos de arranjar uma forma de conseguir arranjar dados para termos a nossa própria base de dados. Tínhamos que fazer com que esta procura fosse automatizada, pois tornava-se insuportável se fossemos nós a ir buscar a três sites, cada um com inúmeras páginas, cada informação sobre cada casa, desperdiçando assim imenso tempo. Posto isto, foi-nos solicitado pelo nosso orientador que utilizássemos Web Crawlers, uma vez que estes ajudariam-nos na parte relativa à recolha de informação dos sites, Workers, estes iam-nos permitir ter a nossa base de dados permanentemente utilizada, pois dependendo da configuração que colocássemos ele correria os nossos Web Crawlers, que por sua vez acediam aos sites, recolhiam as informações e posteriormente armazenavam na nossa base de dados. Para este armazenamento, antes de os dados serem colocados na base de dados, primeiramente eram criados ficheiros JSON provenientes da execução dos Crawlers. Estes ficheiros, por sua vez, eram armazenados tendo por base a tecnologia AWS S3, que consiste num serviço da Cloud Computing oferecido pela Amazon Web Services (AWS). Após os dados estarem devidamente armazenados eram posteriormente inseridos na base de dados.
Após sabermos o que fazer, fizemos uma pesquisa sobre os melhores sites de imobiliário, mas deparamo-nos com dificuldades, pois nem todos os sites nos davam acesso, nem todos tinham os mesmos dados, pelo que haviam colunas que por vezes iriam ficar completamente vazias.

Após isto selecionamos dois sites de vários que encontramos, tendo assim sendo os selecionados:

- Imovirtual – secção da compra de imóveis;
- Imovirtual – secção do aluguer de imoveis;
- IOL – secção da compra de imoveis.

Após análise de todos os requisitos, passamos então para as escolhas das tecnologias.

4.1.1.2 Descrição e enumeração das tecnologias utilizadas

As três tecnologias implementadas para a criação dos Web Crawlers, para a criação dos Workers e armazenamento dos dados são:

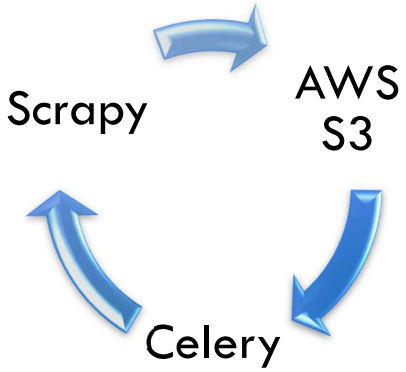


Figura 3. Enumeração das tecnologias usadas

- Web Crawler (Scrapy) – Definições e Implementação

Cada vez mais a tecnologia nos traz benefícios. Muitas vezes o acesso aos dados não é fácil e por mais que desejássemos que tudo estivesse disponível no formato que preferíamos, isso não acontece, pois, os dados são divulgados de forma diferente na internet. Com o uso automatizado de formas de coletar dados web poupa-se imenso tempo e esforço.

As duas formas de coletar dados Web são através de Web Crawler ou de Web Scraping, que correspondem a duas formas de mineração que permitem a extração de dados de sites da web, convertendo-os em informação estruturada para posteriormente analisar.
A forma mais básica de recolher dados é através do download manual de páginas, copiando e colando o conteúdo. Contudo, esta técnica pode ser aperfeiçoada, podendo assim ser feita através de um software que simula uma navegação humana por diversos sites, extraindo informações especificas.

Estas formas de obter dados são muitos semelhantes à indexação web (utilizada pela maioria dos motores de busca), mas a motivação final é muito diferente. A indexação web é usada para ajudar a tornar os motores de busca mais eficientes, já estes mecanismos são tipicamente usados para diferentes razões, como por exemplo, a comparação de preços online.

Os conceitos de Web Crawler e Web Scraping são muito idênticos, pelo que, por vezes, existe uma confusão entre os conceitos, por isso fazemos aqui uma breve introdução sobre os mesmos. Assim sendo um Web Crawler consiste num processo de localizar e pesquisar links da Web de forma iterativa, começando de uma lista de URLs enquanto que o Web Scraping consiste num processamento de um documento da Web e extração de informações a partir dele.

Um Web Crawler é um programa de computador que navega pela rede mundial de uma forma metódica e automatizada. Assim sendo, um Crawler, também conhecido como Spider, consiste num robô usado pelos pesquisadores para encontrar e indexar páginas de um site. Esta captura informações das páginas e regista os links encontrados, possibilitando encontrar outras páginas e mantendo a base de dados atualizada.
Existem ferramentas que facilitam o trabalho dos Crawlers e tornam a indexação das páginas de um site mais rápidas. Estas ferramentas são: *Sitemap.xml*, que consiste num ficheiro que contém uma lista de todas as páginas de um site e ao aceder a essa lista, o Crawler sabe quais as páginas que existem para indexar; a outra ferramenta é *Robots.txt*, que consiste num ficheiro em formato de texto que trabalha como um filtro, avisando os Crawlers de quais as páginas e diretórios que não devem ser indexados.
Os Web Crawlers são principalmente utilizados para criar uma cópia de todas as páginas visitadas para um pós-processamento por um motor de busca que irá indexar as páginas para prover buscas mais rápidas.

O funcionamento de um Web Crawler pode ser verificado por este seguinte esquema:

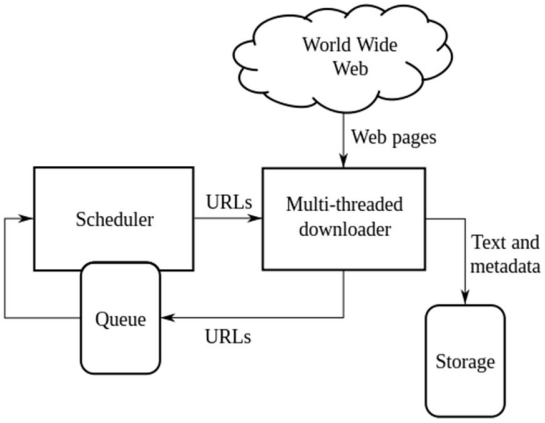


Figura 4. Funcionamento de um Web Crawler

Após observação do esquema, podemos dizer que o funcionamento do Crawler começa com uma lista de endereços (URLs) para visitar, e à medida que o Crawler visita esses endereços, ele identifica todas as ligações na página e adiciona-as na lista de endereços a visitar. Esses tais endereços são visitados recursivamente de acordo com um conjunto de regras definidas pelo criador do Crawler. Este funcionamento aqui mencionado, foi o que ocorreu nos nossos Sites definidos anteriormente.

Apresentado então a definição de Web Crawler e como é o seu funcionamento, passamos então para a explicação sobre a tecnologia selecionada para implementação do mesmo.

O Scrapy é uma framework de rastreamento da Web de código aberto e gratuito, que é escrito em Python. Originalmente projetado para extração de dados na Web, ele também pode ser usado para extrair dados usando APIs.

A arquitetura do projeto Scrapy é construída em torno de "spiders", que são rastreadores autónomos que recebem um conjunto de instruções. Este também fornece um Shell de rastreamento web que pode ser usado pelos desenvolvedores para testar as suas suposições sobre o comportamento de um

site. Este Shell foi-nos bastante útil, pois através dele pudemos testar as nossas suposições, permitindo assim que os nossos Crawlers corressem sem erros, dando-nos as devidas informações, e poupando-nos imenso tempo, uma vez que não tínhamos que voltar ao código desenvolvido para saber o que se passava.

O Scrapy como já foi referido, foi a tecnologia usada para criar os nossos Web Crawlers. A construção dos mesmos foi bastante complicada, pois nem todos os sites tinham as mesmas informações e então tivemos que realizar vários Crawlers de diversos sites para conseguirmos chegar aos dados que temos hoje. Devido à forma como os sites estão organizados, e devido ao facto de posteriormente precisarmos dos dados o mais bem organizados possível, tivemos que optar por termos menos features (ou, seja, menos parâmetros), mas com menos passamos em NULL. Pois posteriormente, como vamos analisar, os campos em NULL iam-nos prejudicar em muito. Mesmo com menos features, não limitamos a quantidade de dados colocados na nossa base de dados, apenas temos menos campos de análise (features), mas imensos dados (chegando aos 8000 dados).

O uso do Scrapy para esta funcionalidade trouxe vantagens, tais como, o facto de este ser rápido e poderoso na escrita das regras para extrair os dados, deixando assim que o Scrapy faça o resto; o facto de ser facilmente extensível, pois pode assumir novas funcionalidade facilmente sem ter que

tocar no núcleo; e por fim, o facto de ser portátil, pois como está escrito em Python, ele tanto funciona em Linux, como em Windows, como em Mac.

Por fim, de referir, que escolhemos o Scrapy pela sua facilidade em criar Web Crawlers, pois como somos iniciantes no que diz respeito a este assunto, achamos por bem não dificultar as coisas. Após esta ferramenta ter sido apresentada pelo nosso orientar, fizemos algumas pesquisas sobre a mesma e achámos que realmente seria a melhor hipótese, pois como é uma tecnologia que é desenvolvida

tendo por base a linguagem Python e como estamos à vontade com essa linguagem, não tínhamos porque não usar.

- Storage File (AWS S3) – Definições

As empresas atuais precisam recolher, armazenar e analisar dados em escalas massivas com simplicidade e segurança, e por isso utilizam o AWS S3 pois consiste num armazenamento de objetos criado para armazenar e recuperar qualquer quantidade de dados de qualquer local. Este serviço foi projetado para oferecer resiliência de quase 100% (~99,999%) e armazena dados para milhões de aplicações usadas por lideres de mercado em todos os setores. O S3 oferece recurso abrangentes de segurança e conformidade que cumprem até os requisitos normativos mais rigorosos. Os clientes, por sua vez, podem gerir de forma flexível dados de otimização de custo, controlo de acesso e conformidade. Por fim, referir também, que a Amazon S3 é o serviço de armazenamento na nuvem que conta com maior suporte no mercado, integrado à maior comunidade de soluções de terceiros e parceiros integrados de sistema.

Após várias pesquisas e pelo que foi descrito anteriormente, achamos que utilizar o AWS S3 para armazenar os nossos dados era uma excelente opção, pois estes disponibilizam várias categorias de armazenamento para atender às necessidades existentes. Contudo após algumas tentativas falhadas optamos por não o utilizar, contudo gostaríamos de o fazer e esta vai ser uma das melhorias que queremos fazer no futuro pois a sua utilização iria enriquecer este trabalho. Em vez disto optamos por armazenar os dados localmente.

- Worker (Celery) – Definições

Após termos os Crawlers a funcionar corretamente, estava na altura de criarmos um Worker, para que de forma automática e diariamente este pudesse correr os nossos Crawlers, mantendo a nossa base de dados atualizada.

Numa primeira fase, deparamo-nos que em Windows, uma forma de conseguir elaborar um Worker poderia ser através do Task Scheduler (Agendador de Tarefas). Este é uma componente do Windows que fornece a capacidade de agendar a inicialização de programas ou scripts em horários predefinidos ou após intervalos de tempo especificados. Após esta tentativa de implementação, deparamo-nos que à hora determinada, era executada uma página que continha a informação do Crawler, mas nada ficava registado na nossa base de dados, pelo que nos apercebemos que aquela não era a solução, pois estaria a falhar em algum lugar.

Visto que as nossas tentativas não foram bem-sucedidas, voltamos às buscas e deparamo-nos que para Windows, uma boa forma de criarmos o Worker era utilizando o Celery.

O Celery consiste numa implementação de uma fila de tarefas para aplicações da Web desenvolvidas em Python, usadas para executar de forma assíncrona o trabalho fora do ciclo de solicitações-respostas do HTTP. Este pode ser utilizado para executar trabalhos em segundo plano numa programação regular. As filas de tarefas e a implementação do Celery são uma das partes mais complicadas de uma pilha de aplicações da Web em Python para se entender.

4.1.2 Base de Dados (MySQL)

4.1.2.1 Análise de requisitos

Após conseguirmos obter os dados que provenientes dos Web Crawlers precisávamos de os inserir numa base de dados. Felizmente o que não nos falta são opções e por isso, numa faze inicial pretendíamos utilizar o MongoDB, que é uma base de dados não relacional, sendo classificada como um programa de banco de dados MySQL, uma vez que esta utiliza documentos semelhantes a JSON como esquemas. Como nunca tínhamos utilizado uma base de dados deste género e como o tempo para a execução do projeto não era tão grande como desejávamos, e devido à complexidade do nosso projeto e a falta de experiência em mexer nestas tecnologias, tínhamos que optar por outro solução que nos trouxesse mais garantias de que tínhamos o projeto pronto a tempo, com muita pena nossa, porque seria ótimo para nós assimilarmos conhecimentos sobre outro tipo de base de dados diferentes do que as que costumamos usar.

Após consenso (entre nós e o nosso orientador), achamos por bem selecionar uma base de dados que já tivéssemos o mínimo conhecimento sobre ela, tendo sido então selecionado o MySQL.

Assim que tínhamos a base de dados selecionada, fizemos algumas pesquisas de como esta funcionaria, e como iriamos trabalhar nela, pois a base de dados com qual tínhamos mais à vontade era da Oracle. Assim sendo, para uma melhor manipulação utilizamos uma das suas interfaces gráficas o MySQL Workbench. Para utilizarmos o *MySQL*, tivemos que instalar um servidor e uma aplicação cliente. O servidor vai ser então o responsável por armazenar os dados, responder às aquisições, controlar a consistência dos dados. Por sua vez, a API comunica com o servidor através do SQL. O servidor deve então ser instalado devidamente e configurado para receber conexões dos clientes, assim sendo, o servidor utilizado foi o *WampServer64*

Em suma, os requisitos utilizados nesta parte do projeto são o MySQL, o MySQL Workbench e o WampServer. Mais à frente iremos explicar detalhadamente cada um deles.

4.1.2.2 Descrição e enumeração das tecnologias utilizadas

Uma base de dados consiste num conjunto de ficheiros relacionados entre si com registos sobre pessoas, lugares ou coisas. São coleções organizadas de dados que se relacionam de forma a criar algum sentido e dar mais eficiência durante uma pesquisa ou estudo. São de vital importância para empresas e há duas décadas que se tornaram uma peça fundamental dos sistemas de informação.

O MySQL é um sistema de gestão de base de dados (SGBD) de código aberto que utiliza a linguagem SQL como interface e usado na maioria das vezes por aplicações gratuitas para gerir as suas bases de dados. Esta utiliza a linguagem SQL pois é a linguagem mais popular para inserir, processar e gerir o conteúdo armazenado na base de dados. É atualmente um dos SGBD mais populares. Entre os utilizadores de base de dados MySQL temos a NASA, a Nokia, a Sony. As características desta base de dados são: a sua portabilidade, pois suporta praticamente qualquer plataforma atual; a sua compatibilidade; o seu excelente desempenho e estabilidade; a sua facilidade no manuseamento; o seu suporte de controle transacional; a suas interfaces gráficas como o MySQL Workbench; entre muitas outras características que esta possui.

Na criação de aplicações web abertas e gratuitas, o conjunto de aplicações usado inclui, um sistema operacional (no nosso caso o Windows), um servidor Web (no nosso caso o WampServer), um SGBD (MySQL) e uma linguagem de programação (a que demos mais incidência em todo o trabalho foi o Python, mas também trabalhamos com outras). Assim, o MySQL é um dos componentes centrais da maioria das aplicações públicas da Internet.

O sistema foi desenvolvido pela empresa sueca MySQL AB e publicado, originalmente, em maio de 1995. Após essa data, a empresa foi comprada pela Sun Microsystems e, em janeiro de 2010, integrou a transação bilionária da compra da Sun pela Oracle Corporation. Atualmente, a Oracle tornou o uso da mesma mais restrito e os desenvolvedores criaram, então o projeto MariaDB para

continuar a desenvolver o código de forma totalmente aberta e gratuita. O MariaDB pretende manter compatibilidade com as versões lançadas pela Oracle.

No MySQL, o principal cliente é a interface gráfica cliente fornecida pela Oracle, denominada *MySQL Workbench*. Através desta pode-se executar consultas SQL, administrar o sistema e modelar, criar e manter a base de dados através de um ambiente integrado. Com esta interface o programador deixa que ter que programar na linha de comandos, que para programadores iniciantes (que é o nosso caso) parece que se torna mais complicado, passando assim a trabalhar numa interface muito mais amigável e que nos permite efetuar múltiplas ações de forma muito mais prática.

Para criação da mesma, utilizamos o MySQL Workbench, pelos inúmeros motivos apresentados anteriormente. Nesta interface criamos a nossa base de dados, à qual denominamos *projeto_final*, e efetuamos a criação de duas tabelas, a *user* e *a projeto_dados_crawler*.

	username	password	email
	David	david	darovi@gmail.com
	Joana	ioana	ioanarmmartins18@gmail.com
	Cristina	cristina	cristina@gmail.com

Tabela 1. Representação da tabela user já com alguns dados inseridos

id	Titulo	Tipo_Negocio	Preco	Tipologia	Localizacao	Tipo_Imovel	Condicao	Numero_Quartos	Numero_Casas_Banho	Certificado_Energetico	Area_Util
1	T1+1 B...	Arrendar	700	T2	Benfica, Lisboa	apartamento	Renovado	2	1	A+	65
2	Aparta...	Arrendar	700	T3	Benfica, Lisboa	apartamento	Renovado	3	1	A+	65
3	T2 Alcâ...	Arrendar	700	T2	Benfica, Lisboa	apartamento	Renovado	2	1	A+	65
4	Aparta...	Arrendar	700	T2	Benfica, Lisboa	apartamento	Renovado	2	1	A+	65
5	T-2+1 ...	Arrendar	700	T2	Benfica, Lisboa	apartamento	Renovado	2	1	A+	65
6	T2 Ala...	Arrendar	700	T2	Benfica, Lisboa	apartamento	Renovado	2	1	A+	65

Tabela 2. Representação da tabela projeto_dados_crawler com alguns dos dados obtidos dos crawlers

A tabela *user* tem duas funções: a primeira para guardar todos os dados de registo dos utilizadores do Web Site; e a segunda serve para a autenticação através do login do utilizador. Quanto à autenticação, muitos foram os obstáculos que nos apareceram, pois é um pouco mais complexo,

devido ao facto de nos obrigar a chamar a tabela *user* e verificar se os dados inseridos pelo utilizador correspondem aos que estão guardados na mesma. De referir que, os dados que se encontram na tabela *user* são criados pelo utilizador quando este se regista na nossa pagina Web, e é depois destes dados que se processa todo o processo de autenticação.

Relativamente à tabela *projeto_dados_crawler*, esta vai conter todos os dados que recebemos dos diversos Web Crawlers, esses dados são inseridos automaticamente sempre que os Web Crawlers correrem. Para este procedimento, o que fizemos foi na parte do Scrapy, escrever pequenos pedaços de código nos seus ficheiros de configuração, ficheiros estes que continham as definições corretas para que a ligação entre ambos pode-se ser estabelecida. Para além da função óbvia que esta tabela têm (armazenamento dos dados oriundos dos Web Crawlers), apresenta mais duas funções essenciais no nosso projeto: a primeira corresponde ao facto de enviar os devidos dados em forma de resposta aos pedidos da API, sendo este processo explicado com mais detalhe mais adiante; a outra funcionalidade corresponde ao facto de esta armazenar os dados que vão servir de suporte para os nossos modelos de previsão, estes dados vão ser armazenados num ficheiro CSV, onde posteriormente os iremos colocar a rodar no Jupyter (ferramenta de desenvolvimento dos métodos de previsão), para então aplicarmos os modelos de previsão.

Resta-nos referir que em todo este processo, tanta criação da base de dados, como na ligação entre a base de dados e o Scrapy, o servidor WampServer foi imprescindível. Este consiste num software que efetua a instalação automática de um conjunto de softwares no computador, de modo a facilitar a configuração de um software interpretador de scripts local e uma base de dados no sistema Windows. Resumindo, este consiste numa ferramenta que auxilia no desenvolvimento, permitindo aos programadores de websites testarem o trabalho nos seus próprios computadores, sem necessitarem de acesso à internet.

4.1.3 MÉTODOS

4.1.3.1 Análise de requisitos

Uma das partes essenciais do projeto, e a base na qual o projeto se inseria, era na criação de modelos de previsão do preço das casas. Como já tínhamos uma base de conhecimentos de funcionamento dos métodos, e devido a orientações do nosso orientador, foi-nos fácil escolher os métodos com os quais iriamos trabalhar. Inicialmente escolhemos três, mas devido a contratempos, realizados apenas dois métodos.

Para a realização destes métodos, tendo por base a linguagem Python (neste caso foi utilizada a versão 3) pois como é uma linguagem rica em grandes bibliotecas que tratam e implementam os algoritmos mais usados no Machine Learning, é então praticamente inevitável o seu uso da mesma, visto que esta é a linguagem mais usada para estes tipos de desenvolvimentos.

Após eleita a linguagem, e seleção do Jupyter Notebook como o nosso editor de texto, passámos então para a decisão da escolha dos métodos. Assim, os métodos selecionados foram o Random Forest e a Regressão Linear.

As bibliotecas utilizadas na implementação dos métodos são: Skylearn, Pandas, Numpy, Matplotlib, Seaborn, Statsmodels, Pickle, Requests.

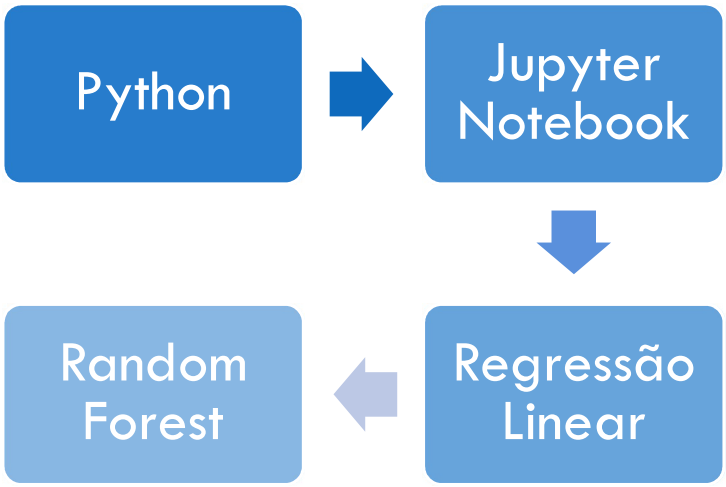


Figure 5. Requisitos necessários para implementação dos métodos

4.1.3.2 Descrição e enumeração das tecnologias utilizadas

Para podermos aplicar um método, os dados primeiramente têm que estar devidamente tratados, para que aplicação do mesmo corra o melhor possível.

Posto isto, devido à forma como obtemos os dados oriundos dos Crawlers, devido à dificuldade que obtivemos em conseguir recolher os dados todos no formato indicado e como se trata do ramo imobiliário, tivemos dois grandes problemas nos nossos dados.

Um dos problemas que tínhamos no nosso dataset devido ao formato dos dados que já referimos anteriormente, é relativo à feature Localização, uma vez que esta não se encontra da maneira correta para que posteriormente a pudéssemos usar. Para solucionarmos este problema fizemos um replace dos dados. Com os dados originais a previsão não era tão precisa, pois os dados eram muito dispersos, então a nossa ideia foi agrupar as localizações por regiões maiores que abrangessem de uma maneira geral, assim em vez de termos vários dados com inúmeras freguesias, colocamos os dados mais compressos colocando-os por concelhos para que quando fossemos realizar as previsões o nosso modelo foi mais preciso. Um exemplo que podemos dar da solução que realizamos é o seguinte:

Figure 6. Exemplificação da resolução do problema

Este condicionamento vai originar uma diminuição do erro de previsão, uma vez que vai existir uma maior probabilidade de haver mais casas em Vila Franca de Xira do que em apenas uma das duas opções que foram mencionadas no exemplo, originando com que o nosso modelo fique o mais bem treinado possível, produzindo boas previsões.

O outro problema mencionado era relativo ao nosso dataset conter dados de casas para arrendar e dados de casas para comprar. Este problema interferia com os valores que posteriormente iriamos obter, e não fazia sentido deixarmos os dados daquela maneira, pois as leituras que íamos obter não era as mais corretas. Então a decisão que tomamos em relação a este aspeto, foi dividir os dados em dois datasets diferentes, ou seja, em vez de um único dataset passamos a ter dois, um que continha os dados unicamente de arrendamentos e outro que continha unicamente os dados de venda. Desta maneira na altura de aplicarmos os métodos e sabermos os resultados, tudo bateria em condições.

Após estas alterações, mais problemas foram surgindo relativamente aos dados. Os modelos só são desenvolvidos e aplicados com inteiros ou Floats e alguns dos nossos dados estavam em formato String. Para além de termos o problema mencionado, também tínhamos que lidar com a falta de dados, ou seja, com os valores NaN, pois como já referimos muitas vezes, nem todos os sites tinham todas as informações que desejamos e, portanto, é normal que isto tenha acontecido.
Após algumas pesquisas e conhecimentos anteriores, concluímos que as variáveis categóricas (as que estão representadas pelo valor String) têm que ser codificadas como valores numéricos. Estas foram transformadas tendo por base o OneHotEncoder (OHE). Com este, uma nova variável é criada para cada valor de uma variável categórica. Antes de aplicarmos o OHE ao nosso dataset, tratamos primeiro dos nossos campos que têm valor NaN. Assim, de referir, que os valores ausentes podem ser bastante significativos e vale a pena investigar o que eles representam. Na nossa solução, o que basicamente fizemos foi substituir os valores NaNs por zero. Isto é feito devido à seguinte linha de código que desenvolvemos:

```
categoricals_arrendar = []
for col, col_type in dataset_arrendar_final.dtypes.iteritems():
     if col_type == 'O':
          categoricals_arrendar.append(col)
     else:
          dataset_arrendar_final[col].fillna(0, inplace=True)
```

Figura 7. Linha de código corresponde ao tratamento de dados

Como podemos observar pelo código descrito na Figura 7, irá ser iterado todas as colunas do nosso dataset e anexar variáveis categóricas à lista de categorias e às variáveis não categorias irá simplesmente substituir os valores NaNs por zeros. Sabemos que ao preencher NaNs com um único valor podemos ter consequências indesejáveis, especialmente se o valor que estamos a substituir estiver dentro do intervalo usado para a variável numérica. Por isso, antes de aplicarmos esta decisão tentamos muitas outras soluções, mas muitas tinham por base a eliminação de linhas, o que ia fazer com que tivéssemos menos dados para o nosso modelo e que por consequência, as nossas previsões não pudessem ser tão precisas, por isso decidimos correr o risco e optar pela solução mencionada.

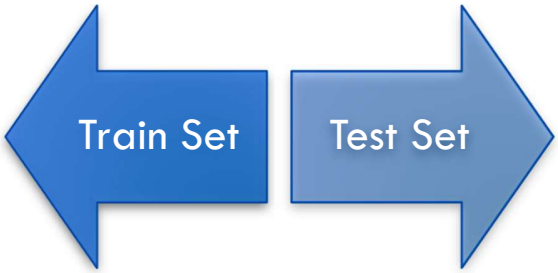
Uma vez resolvido o problema relativo à falta de valores, o nosso dataset está pronto para o OHE das nossas variáveis categóricas. O Pandas fornece um método simples, denominado *get_dummies* para criar variáveis OHE para um dado dataframe. Assim, com base nesse método, desenvolvemos a seguinte linha de código:

```
#one hot encode
dataset_ohe_arrendar = pd.get_dummies(dataset_arrendar_final, columns=categoricals_arrendar, dummy_na=True)
```

Figura 8. Representação do código para OHE

O bom do OHE é que este é determinista, ou seja, uma nova coluna é criada para cada combinação de coluna/valor, no formato *column_value*. Assim, todas as features que eram do tipo String, como é o caso por exemplo da feature *Tipo_Imovel*, os dados com o OHE passaram a ser do género: *Tipo_Imovel_Apartamento, Tipo_Imovel_Penthouse.*

Realizado o tratamento dos dados, podemos então partir para a parte de treinar o modelo. Os dados são divididos através de uma função Random, em dois campos:



No train set, iram estar colocados 80% dos dados do nosso dataset, e tal como o nome indica serve para treinar o modelo com o objetivo de o aperfeiçoar, para posteriormente executá-lo. Por outro lado, no test set, iram estar os restantes 20% dos dados, e este serve para verificar a precisão do modelo.

Em suma, o conjunto de dados de treino é usado para preparar um modelo para executá-lo. Pretendemos que o conjunto de dados de treino seja novo, e que os valores de saída sejam retidos do algoritmo. Reunimos as previsões do modelo de treino nas entradas do dataset de teste e comparamos aos valores de saída retidos do conjunto de testes. A comparação de previsões e saídas retidas no conjunto de dados de teste permite-nos incluir uma medida de desempenho para o modelo no conjunto de dados de teste. Esta é uma estimativa da habilidade do algoritmo treinado do problema ao fazer previsões sobre dados não vistos.

Passada então a fase de treino de dados, podemos então aplicar os nossos dois modelos selecionados, são eles: a Regressão Linear e o Random Forest.

Antes de começarmos a explicar os métodos, queríamos referir que existem vários tipos de modelos, mas os que aqui vamos dar mais enfase são os seguintes:

Os modelos de regressão são uma subcategoria de aprendizagem supervisionada usada quando o valor que está a ser previsto difere de um "sim ou não" e que siga um espectro contínuo. Sistemas de regressão poderiam ser usados, por exemplo, para responder às perguntas: "Quanto custa?" ou "Quantos existem?".

Por outro lado, os modelos de classificação, também são uma subcategoria de aprendizagem supervisionada Classificação é o processo de tomar algum tipo de entrada e atribuir um rótulo a ela. Sistemas de classificação são usados geralmente quando as previsões são de natureza distinta, ou seja, um simples "sim ou não". Exemplo: Mapeamento de uma imagem de uma pessoa e classificação como masculino ou feminino.

Assim sendo, faltamos apenas referir no que consiste a aprendizagem supervisionada, para que melhor possamos entender os coisas. Assim, a aprendizagem supervisionada é o termo usado sempre que o programa é "treinado" sobre um conjunto de dados pré-definido. Baseado no treinamento com os dados pré-definidos, o programa pode tomar decisões precisas quando recebe novos dados. Um exemplo que podemos dar é: pode-se usar um conjunto de dados de recursos humanos para treinamento da Machine Learning, que tenha tweets marcados como positivos, negativos e neutros e assim treinar um classificador de análise de sentimento.

Podemos então referir, que os dois métodos selecionados, um é de regressão, que tal como o nome indica é o método Regressão Linear, e o outro é de classificação, que corresponde ao Random Forest. Passamos então à descrição detalhada de cada um deles.

❖ Modelo de Regressão (Regressão Linear) – Definição e Implementação

A Regressão Linear é um dos modelos mais conhecidos em estatística e Machine Learning. A representação é uma equação linear que combina um conjunto específico de valores de entrada (x) cuja solução é a saída prevista para esse conjunto de valores de entrada (y). Como tal, os valores de entrada (x) e o valor de saída são numéricos. Esta equação atribui um fator de escala a cada valor ou coluna de entrada, chamado de coeficiente e representado pela letra maiúscula Beta.

Existem varias técnicas para se usar a regressão linear, porque este método foi devidamente estudado. Existe assim, a regressão linear simples que tem um input e podemos usar estatísticas para estimar os coeficientes. Contudo requer o calculo das propriedades estatísticas dos dados e todos eles devem estar disponíveis para serem calculados. No cenário em que existe um ou mais input é necessário usar um processo de otimização dos valores dos coeficientes, minimizando iterativamente o erro do modelo em seus dados de treino, tal pode ser feito através da operação Gradient Descent. O Gradient Descent funciona através de valores aleatórios para cada coeficiente. A soma dos erros quadrados é calculada para cada par de valores de entrada e saída. Uma taxa de aprendizagem é usada como um fator de escala e os coeficientes são atualizados no sentido de minimizar o erro. Este processo é repetido até que o erro mínimo de soma quadrática seja alcançado ou que nenhuma melhoria adicional seja possível.

Na regressão linear, os dados têm que ter uma relação entre input e output linear. Os dados não devem ter ruídos. Por isso, deve-se considerar o cálculo de correlações entre pares para os dados de input e a remoção dos correlacionados. Se as variáveis de input e output tiverem uma distribuição gaussiana, o algoritmo fará previsões mais confiáveis. Além disso se redimensionarmos as variáveis de input usando padronização ou normalização a previsão também será mais confiável.

Após esta breve descrição sobre o método, temos a referir que ao aplicarmos o método aos nossos dados o valor de score. O valor de score mencionado pode ser interpretado como uma média ponderada da precisão, onde se atinge o seu melhor valor quando o valor de aproxima de 1 e o seu pior valor quando se aproxima de 0. Para termos as melhores previsões possíveis, decidimos colocar o objetivo do score nos 85%, visto que é um valor aceitável para as previsões.
Ao aplicarmos todo o procedimento do método nos nossos dados, visualizamos que o nosso score era abaixo dos 85%, rondando valores entre os 60 -70%, pelo que concluímos logo que muito provavelmente as previsões não iriam sair as mais corretas possíveis, pois a margem de erro era alta e então o valor que íamos passar ao utilizador não era o mais correto, como podemos observar nas seguintes tabelas:

	Erro	Previsao	Real
799	790.213295	2790.213295	2000
731	42.266870	2742.266870	2700
651	-209.861371	990.138629	1200
41	-31.250530	718.749470	750
730	314.661118	2064.661118	1750

Tabela 3. Representação dos cinco primeiros valores de previsão das casas para arrendar

	Erro	Previsao	Real
5001	-75543.642804	5.744564e+05	650000
2935	-572669.611238	1.177330e+06	1750000
4000	43769.846330	3.736698e+05	329900
4376	50201.586607	1.902016e+05	140000
5508	-176841.042343	7.131590e+05	890000

Tabela 4. Representação dos cinco primeiros valores de previsão das casas para comprar

Ao analisarmos as Tabelas 3 e 4, podemos então verificar o que referimos anteriormente, para além de muitos valores negativos, muitos deles altíssimos, também temos valores positivos que também são bastante elevadíssimos, pelo que o contraste entre os preços é enorme. Dado que temos os valores acima referidos, a nossa opção foi não aplicar este método como previsões e passamos então para a aplicação do seguinte.

Por fim, de referir, que este modelo tem como desvantagem apenas considerar relacionamentos lineares, tomar como base a média da variável dependente e ser sensível a outliers.

❖ Modelo de classificação (Random Forest) – Definição e Implementação

Este algoritmo é um dos algoritmos mais utilizados, devido à sua simplicidade e pelo facto de poder ser usado tanto para tarefas de classificação como de regressão. Funciona com a criação uma floresta e torna-a aleatória. Essa floresta, é um conjunto de Árvores de Decisão, que na maioria das vezes são treinadas com o método "bagging", que é uma combinação de modelos de aprendizagem que aumenta o resultado geral. Por outras palavras cria várias árvores de decisão e junta-as para obter uma previsão ainda mais precisa e estável. Na figura a baixo podemos ver como seria uma floresta aleatória com duas árvores:

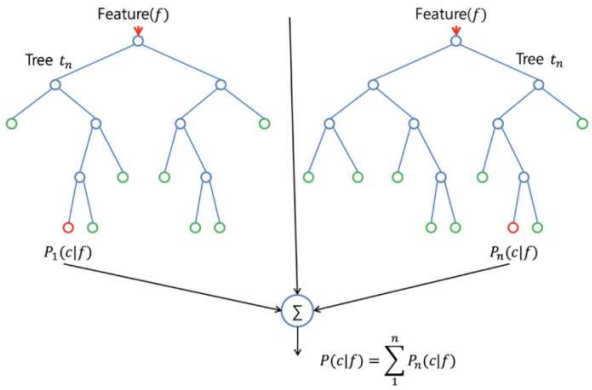


Figura 9. Representação das arvores realizadas pelo método

O Random Forest tem quase os mesmos hiperparâmetros que uma árvore de decisão ou um classificador de bagging. Este adiciona mais aleatoriedade ao modelo, enquanto as árvores crescem. Em vez de procurar a feature mais importante ao dividir um nó, ele procura a melhor feature entre um subconjunto aleatório de features. Isso resulta em uma ampla diversidade que geralmente resulta num modelo melhor. Portanto apenas um subconjunto aleatório das features é levado em consideração pelo algoritmo para dividir um nó.

Outra grande qualidade deste algoritmo é que é muito fácil medir a importância relativa de cada feature na previsão, pois o Sklearn fornece uma ótima ferramenta que mede a importância de uma feature, observando o quanto os nós da árvore, que usam essa feature, reduzem a impureza em todas as árvores da floresta. Ele calcula essa pontuação automaticamente para cada feature após o treino e dimensiona os resultados, de modo que a soma de toda a importância seja igual a 1.

Uma das grandes diferenças entre o Random Forest e uma Decision Tree é que se for inserido um conjunto de dados de treino com features e labels em uma Decision Tree, vai formular um conjunto de regras, que serão usadas para fazer as previsões. Enquanto, o Random Forest seleciona aleatoriamente observações e features para construir várias árvores de decisão e, em seguida, calcula a média dos resultados.

Uma maneira fácil de aumentar a precisão do Random Forest é aumentando o número de árvores, uma vez que aumenta o desempenho e torna as previsões mais estáveis, mas também diminui a velocidade do cálculo. Isso é conseguido através do hiperparâmetro *n_estimadors*.

A principal limitação da Random Forest é que um grande número de árvores pode tornar o algoritmo lento e ineficaz para previsões em tempo real. Em geral, este algoritmo é rápido de treinar, mas muito lento para criar previsões depois de treinado. Uma previsão mais precisa requer mais árvores, o que resulta em um modelo mais lento. Na maioria das aplicações do mundo real, o algoritmo é rápido o suficiente, mas certamente pode haver situações em que o desempenho em tempo de execução é importante e nesses casos deve-se usar outro algoritmo.

Este método foi sem dúvida o que aplicamos para fazer as previsões dos preços das nossas casas. Devido a tudo o que foi referido anteriormente sobre este, conseguimos afirmar que o modelo realmente é bastante eficaz em termos de previsões. Conseguimos comprovar isso pelo score que obtivemos ao correrm os este método nos nossos dados.

O score obtido com todos os dados do nosso dataset rondava os 95%, muito acima do objetivo estabelecido, e com um score tão alto é de prever que as previsões estejam muito próximo dos preços reias. Podemos então confirmar esta afirmação, através da observação das seguintes tabelas:

	Erro	Previsao	Real
799	1.642036	2001.642036	2000
731	0.000000	2700.000000	2700
651	0.000000	1200.000000	1200
41	-0.041051	749.958949	750
730	3.119869	1753.119869	1750

Tabela 5. Representação dos cinco primeiros valores de previsão das casas para arrendar

	Erro	Previsao	Real
5001	0.000000	6.500000e+05	650000
2935	-8.852169	1.749991e+06	1750000
4000	0.000000	3.299000e+05	329900
4376	0.000000	1.400000e+05	140000
5508	292.800236	8.902928e+05	890000

Tabela 6. Representação dos cinco primeiros valores de previsão das casas para comprar

Ao observarmos as Tabelas 5 e 6, podemos então constatar que estamos perante um método bastante bom para fazer previsões, visto que os valores de erro são mínimos. Mesmo apresentado valores mínimos, e outros valores mais altos, é normal devido aos outliers que temos no nosso modelo.

Após conclusão de que os nossos valores de previsão estão bastantes bons, decidimos efetuar uns gráficos para que de uma forma visual, se entenda melhor os nossos valores. Assim, observando os seguintes gráficos:

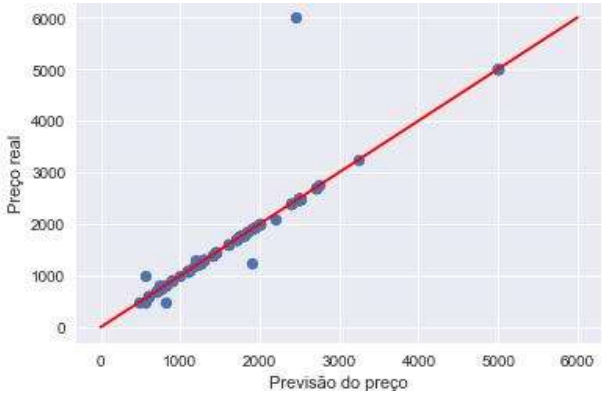


Figura 10. Representa um contraste entre os preços previstos e os preços reais das casas para arrendar

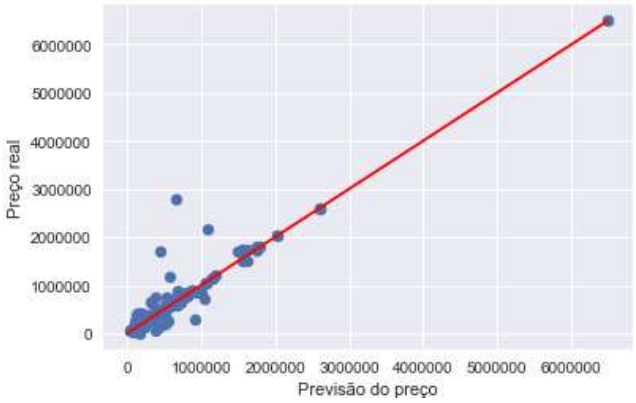


Figura 11. Representa um contraste entre os preços previstos e os preços reais das casas para comprar

Analisando os gráficos, podemos referir que a linha vermelha representa as previsões perfeitas, ou seja, as previsões corretas, e por isso, quanto mais próximo os pontos azuis estiverem perto da linha vermelha mais correta será a previsão. Se todas as previsões fossem perfeitas então todos os pontos azuis estariam em cima da linha vermelha. Nestes gráficos podemos reparar que temos alguns outliers que representam os preços de casas que eram demasiados altos ou demasiado baixos para as suas caraterísticas, e por isso temos aqueles valores que estão mais dispersos. De resto podemos reparar que os pontos azuis estão muito próximos da linha vermelha, o que evidência tudo o que referimos anteriormente.

Para termos uma visão mais ampla do sucesso da previsão dos nossos métodos podemos calcular o erro RMSD **(root-mean-square deviation)** ou RMSE **(root-mean-square deviation)**. Este consiste numa medida frequentemente usada das diferenças entre valores (amostra) previstos por um modelo ou estimador e os valores observados. De referir também que o RMSD representa a raiz quadrada do segundo momento da amostra das diferenças entre os valores previstos e os valores observados. Para o nosso caso só nos interessa a relação do RMSE com o Machine Learning e neste contexto o RMSE mede o desvio padrão das previsões da verdade fundamental. Esta é a relação entre o RMSE e a classificação. O RMSE é uma maneira de medir o desempenho de um classificador, já a taxa de erro mede a percentagem que com que o modelo erra a previsão.

Assim sendo o RMSE e erro do nosso modelo, são de:

RMSE: 0.05601

Tx. Erro: 0.083

A leitura que podemos retirar deste erro é que, o desvio normal do valor real vai ser de e^0.05601 que vai ser igual a 1,057.

4.1.3.3 Análise de dados

Para melhor compreensão dos dados contidos no nosso dataset, decidimos criar alguns gráficos. Estes ajudaram-nos a ter uma ideia mais clara, sendo assim mais fácil redigirmos alguns pontos interessantes dos nossos dados, tais como a relação do preço das casas com a área útil ou com a tipologia. Assim, passamos à demonstração de alguns gráficos seguidos da sua leitura.

Primeiro, fomos descobrir qual seria a feature com maior impacto na oscilação dos preços e para isso utilizamos o método abs() e obtivemos os seguintes resultados:

```
Area_Util            0.792098
Numero_Casas_Banho   0.535092
Condicao_Novo        0.309633
```

Outra forma para conseguirmos observar a relação entre as features, foi através do seguinte gráfico colocado adiante, onde podemos ver a relação de todas as nossas features de interesse. Assim sendo, no seguinte gráfico:

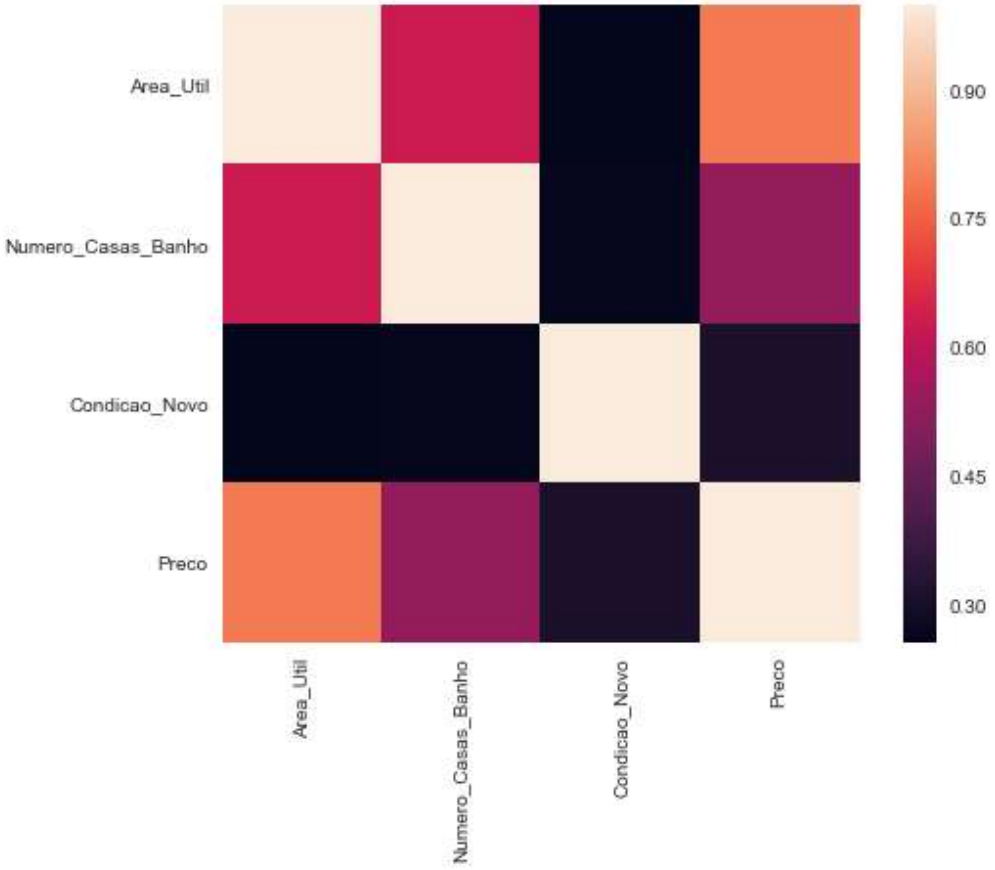


Figura 12. Heatmap

Afirmamos que cada uma das features analisadas influenciam-se mutuamente. A análise deste gráfico é realizada através de cores, sendo que quanto mais claro for a cor maior a influência que uma determinada feature tem com a outra. Assim, podemos concluir que o preço e a área útil têm uma grande influencia, enquanto que em contrapartida as features Condição_Novo e o Numero_Casas_Banho pouco ou nada se influenciam.

Ao observarmos a figura colocada anteriormente, podemos concluir que a feature que tem maior impacto no preço das casas é a feature *"Area_Util"*. Após esta descoberta, passamos então para a implementação do gráfico, o qual obtivemos o seguinte:

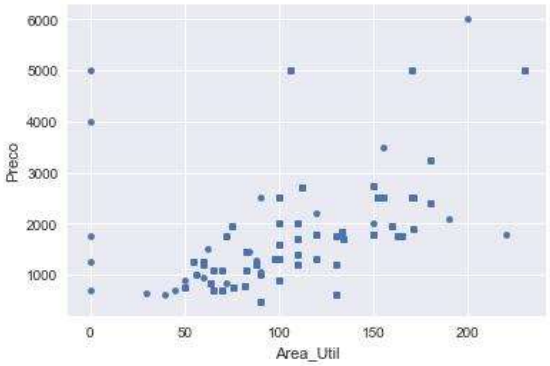


Figura 13. Relação entre Area Util e Preço das casas para arrendar

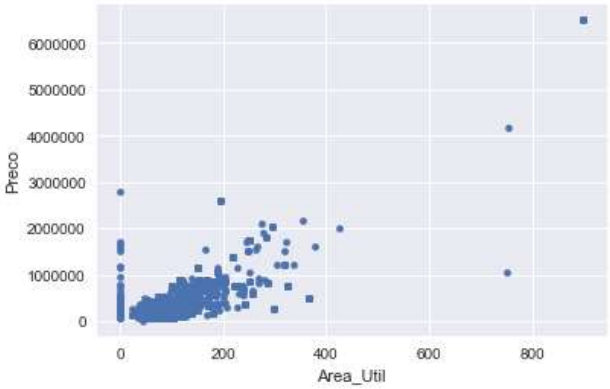


Figura 14. Relação entre Area Util e Preço das casas para compra

Observando o gráfico da figura 8 podemos evidenciar que a área dos imóveis está em média situada entre os 0-200 m^2 na área de venda, e entre os 50- 200 m^2 na área de arrendamento, sendo que estes são os imóveis que têm um preço mais reduzido, demonstrando assim a relação proporcional entre o preço e a área, visto que quanto maior a área maior será o preço do imóvel, verificando-se assim que na região de Lisboa (onde incide a nossa amostra do projeto) os imóveis têm maioritariamente áreas reduzidas contudo os preços nesta região são bastante elevados não devido à área, mas sim devido à localização, ou seja, as casas na região de lisboa têm como referido anteriormente, preços bastante elevados sendo que podem chegar a mais do que 6000000€ consoante a área apresentada. Deste modo podemos realçar que uma casa de 200 m^2 pode custar na região de lisboa cerca de 1000000€, aumentando progressivamente o preço consoante o aumento da área da casa sempre acompanhada da localização da habitação.

Contudo, e observando a grande mancha azul que o gráfico apresenta, podemos observar que as casas com dimensões reduzidas(0-200m^2) apresentam preços bastante elevados proporcionalmente à sua área e nunca muito dispares do preço mais reduzido (1000000€).

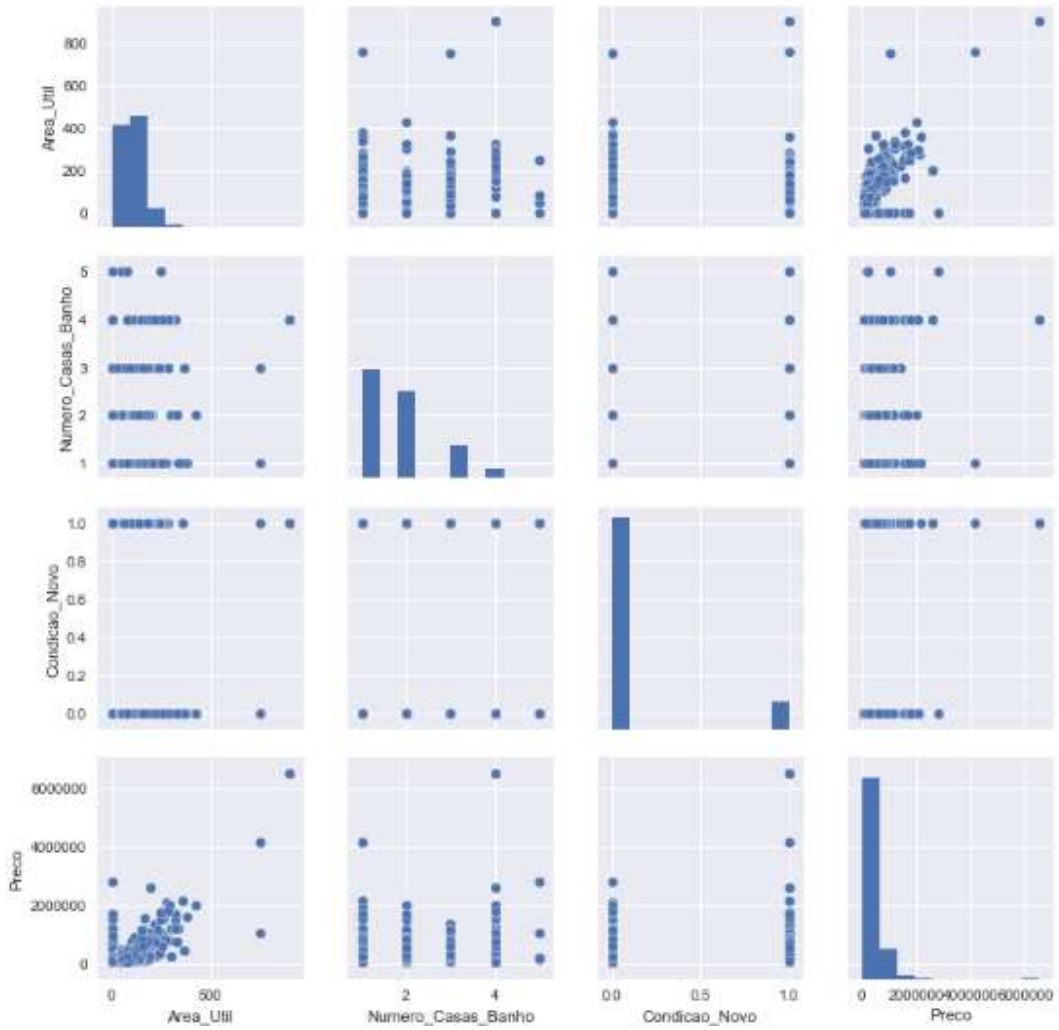


Figura 15. Gráficos que ilustram a relação entre features

Tal como podemos observar, o único que temos clara evolução do preço é a área útil, se observarmos a relação preço número de casas de banho verificamos que ter 1 ou 3 casas de banho pode não influenciar o preço da casa. Já a condição novo, que representa se a casa é nova ou não, já tem influencia no preço como podemos ver as casas mais caras são novas apesar de termos casas que não são novas com um preço caro também. Outro dado interessante é que o número de casas

de banho não aumenta com o a área útil, se observarmos as casas com 5 casas de banho têm menos de 500 metros quadrados de área útil.

4.1.4 RESTFUL API

4.1.4.1 Análise de requisitos

Após termos todos os dados e os respetivos métodos a trabalhar, precisávamos de uma maneira de os apresentar e para isso foi nos requisitado a criação de uma API (Interface de programação de aplicativos). Existem várias tecnologias para criar um API e como nunca tínhamos mexido em nenhuma, pedimos ao nosso orientador que nos aconselhasse. Assim sendo a nossa API de eleição foi e o Flask, que utiliza a linguagem Python em Backend.

Para desenvolvimento do código, utilizamos como editores de texto o Visual Sudio Code e o Sublime.

Uma vez que a API era escrita em Python, tínhamos que usar a distribuição Anaconda para que pudéssemos correr o código, sendo este um dos outros requisitos.

Numa primeira fase, a nossa API estava unicamente conectada com a nossa base de dados, e para que esta funcionasse, para além de várias instalações que tivemos que efetuar no Anaconda, tínhamos também que utilizar bibliotecas, tais como a MySQLdb ou a pymsql.

Em suma, as tecnologias utilizadas para a criação da Restful API, essencialmente Flask e dois editores de texto. Mais adiante, iremos então explicar com mais detalhe como todo este processo se desenvolveu.

4.1.4.2 Descrição e enumeração das tecnologias utilizadas

Uma API é criada quando uma empresa de software tem a intenção de que outros criadores de software desenvolvam produtos associados ao seu serviço. Através das APIs, as aplicações podem comunicar-se uns com os outros sem conhecimento ou intervenção dos utilizadores. Elas funcionam através da comunicação de diversos códigos, definindo comportamentos específicos de determinado objeto em uma interface. A API liga as diversas funções em um site de maneira que possam ser utilizadas em outras aplicações. Ou seja, é através da criação da API, que poderíamos adapta--la para servir varias Web Pages e/ou aplicações mobile.

Uma das grandes vantagens de usar uma API é a compatibilidade entre linguagens, pois qualquer aplicação que estabeleça uma ligação com a API pode fazer-lhe pedidos. Ou seja, é possível ter uma aplicação IOS e outra aplicação Android a serem servidos pela mesma API.

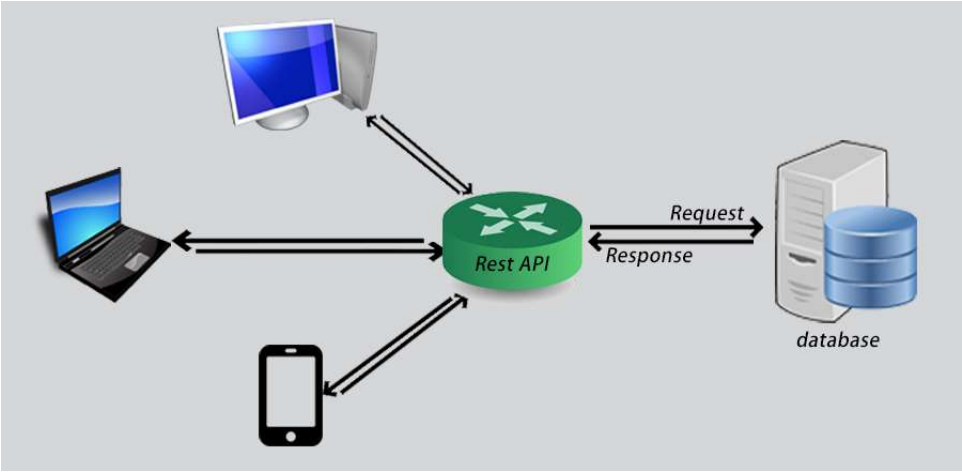


Figura 16. Representação do funcionamento de uma Rest API

A ideia por trás da API, é fazer com que seja a própria a efetuar pedidos diretamente à base de dados.

Suponhamos que temos apenas uma Web Page e uma base de dados, quando o utilizador utiliza essa Web Page para pesquisar casas em Arroios, por exemplo, vai ser a Web Page a fazer o pedido diretamente na base de dados. Contudo se tivermos uma API no meio, o Web Page passa a fazer um pedido à API e vai ser a API que vai fazer o pedido à base de dados, não havendo assim um contacto direto entre a Web Page e a base de dados.

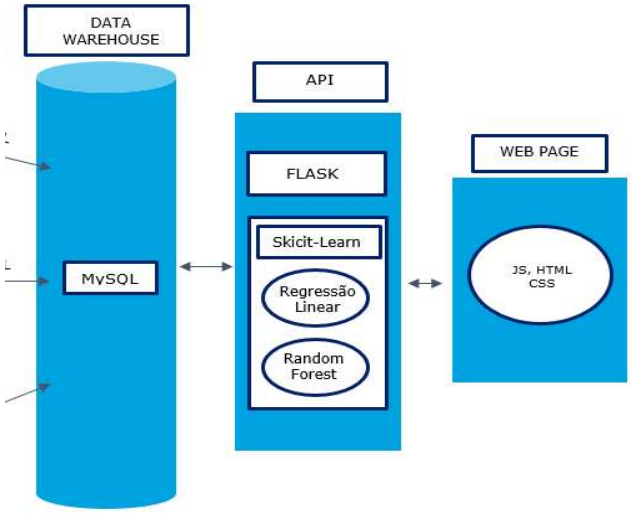


Figura 17. Representação de uma parte do modelo do projeto

Observando o esquema acima indicado, podemos entender melhor o papel de uma API. Como se pode observar, em nenhuma ocasião a Web Page envia diretamente pedidos à base de dados, uma vez que só se pode efetuar pedidos à API. Esta por sua vez vai efetuar um pedido à base de dados e vai receber posteriormente uma resposta da base de dados com a informação que pretende. Após esta conter a devida informação passada pela base de dados, esta vai enviar a informação à Web Page. O mecanismo aqui descrito foi o mecanismo aplicado na nossa aplicação.

Os pedidos feitos à API são pedidos HTTP realizados através de dois métodos: o POST e o GET. Para resolver os problemas relacionados à semântica, quando as requisições HTTP eram feitas, foi criado o uso de novos métodos HTTP, que permitiram o uso do HTTP de uma forma muito mais próxima da realidade. Com a criação desses métodos criou-se o REST(Transferência de Estado Representacional),em que este consiste em princípios/regras/constraints que, quando seguidas, permitem a criação de um projeto com interfaces bem definidas, o que leva a que aplicações se possam comunicar.

A API que construímos e que nos foi solicitada, era uma API RESTFUL, sendo então uma API baseada na tecnologia REST, que é uma abordagem para comunicações frequentemente usadas no desenvolvimento de serviços da web tal como foi falado anteriormente. A API RESTful funciona dividindo uma transação para criar uma série de pequenos módulos. Cada módulo aborda uma parte subjacente específica da transação. Essa modularidade fornece aos desenvolvedores muita flexibilidade.

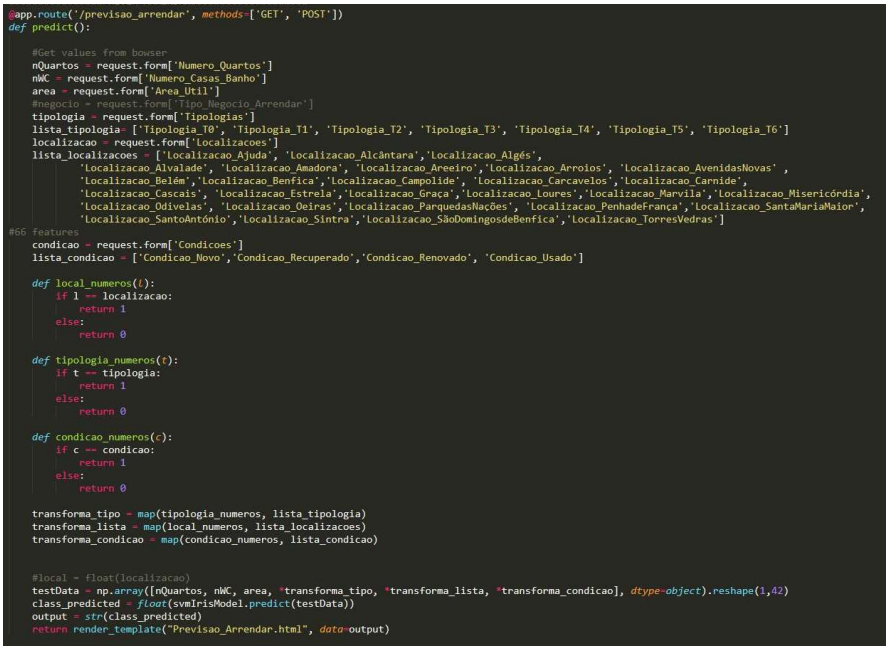
Um pequeno exemplo de como construímos a nossa API tendo por base a Framework Flask, é o seguinte:

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"


if __name__ == '__main__':
    app.run(debug=True)
```

Figura 18. Pequena demonstração de código de desenvolvimento de Flask

O código mencionado acima, é o exemplo geral dado em maior parte dos sites, para iniciantes em Flask.

Após vários tutoriais, começamos a criar as nossas próprias funções e podemos até enumerar uma para que fique aqui constado a estrutura que criamos. Damos como exemplo a função criada para a previsão das casas para aluguer:



```
@app.route('/previsao_arrendar', methods=['GET', 'POST'])
def predict():

    #Get values from bowser
    nQuartos = request.form['Numero_Quartos']
    nWC = request.form['Numero_Casas_Banho']
    area = request.form['Area_Util']
    #negocio = request.form['Tipo_Negocio_Arrendar']
    tipologia = request.form['Tipologias']
    lista_tipologia= ['Tipologia_T0', 'Tipologia_T1', 'Tipologia_T2', 'Tipologia_T3', 'Tipologia_T4', 'Tipologia_T5', 'Tipologia_T6']
    localizacao = request.form['Localizacoes']
    lista_localizacoes = ['Localizacao_Ajuda', 'Localizacao_Alcântara','Localizacao_Algés',
            'Localizacao_Alvalade', 'Localizacao_Amadora', 'Localizacao_Areeiro','Localizacao_Arroios', 'Localizacao_AvenidasNovas' ,
            'Localizacao_Belém','Localizacao_Benfica','Localizacao_Campolide', 'Localizacao_Carcavelos','Localizacao_Carnide',
            'Localizacao_Cascais', 'Localizacao_Estrela','Localizacao_Graça','Localizacao_Loures','Localizacao_Marvila','Localizacao_Misericórdia',
            'Localizacao_Odivelas', 'Localizacao_Oeiras','Localizacao_ParquedasNações', 'Localizacao_PenhadeFrança','Localizacao_SantaMariaMaior',
            'Localizacao_SantoAntónio','Localizacao_Sintra','Localizacao_SãoDomingosdeBenfica','Localizacao_TorresVedras']
#66 features
    condicao = request.form['Condicoes']
    lista_condicao = ['Condicao_Novo','Condicao_Recuperado','Condicao_Renovado', 'Condicao_Usado']

    def local_numeros(l):
        if l == localizacao:
            return 1
        else:
            return 0

    def tipologia_numeros(t):
        if t == tipologia:
            return 1
        else:
            return 0

    def condicao_numeros(c):
        if c == condicao:
            return 1
        else:
            return 0

    transforma_tipo = map(tipologia_numeros, lista_tipologia)
    transforma_lista = map(local_numeros, lista_localizacoes)
    transforma_condicao = map(condicao_numeros, lista_condicao)


    #local = float(localizacao)
    testData = np.array([nQuartos, nWC, area, *transforma_tipo, *transforma_lista, *transforma_condicao], dtype=object).reshape(1,42)
    class_predicted = float(svmIrisModel.predict(testData))
    output = str(class_predicted)
    return render_template("Previsao_Arrendar.html", data=output)
```

Figura 19. Exemplificação do código realizado para a previsão em Flask

Para todas as funções que a nossa API tem seguimos sempre a logica acima indica.

Outro fator que era exigido na nossa API era a autenticação, ou seja, os utilizadores só podem ter acesso aos dados fornecidos pela mesma caso tivessem uma conta. Caso não tivessem uma conta, então tínhamos

um campo onde se podia efetuar o registo, para que estes futuramente possam então fazer o login na página e navegar nela, procurando pelos imoveis que desejarem.

A parte da autenticação foi um campo onde tivemos bastantes dificuldades, pois nunca tínhamos trabalho com o Flask, então tornou-se um bocado mais difícil a procura pelas ferramentas que este proporcionava para termos a maior segurança possível no nosso site.

Após múltiplas buscas, deparamo-nos que o Flask tinha uma componente Login para implementação da autenticação e gestão de sessão. A biblioteca traz: armazenamento do ID do utilizador logado na sessão e permite o login e o logout; permite restringir views para logar ou "deslogar" utilizadores; lida com o
normalmente complicado "remembre me"; ajuda a proteger a sessão do utilizador prevenindo o vazamento de valores de cookies; e integra-se com Flask-Principal ou outras extensões para autorização.

O Flask Login possui alguns decorators para implementação cobrindo mais de uma maneira de autenticação: *user_loader*, para o código saber como carregar um usuário logado da sessão; *request_loader*, que recebe um request do Flask para implementação da autenticação.

A seguinte imagem, dá um exemplo de como efetuar a implementação do Flask Login:

```
login_manager = flask_login.LoginManager()

login_manager.init_app(app)

class User(flask_login.UserMixin):
    pass

@login_manager.user_loader
def user_loader(girl_id):
    return girl_by_id(girl_id)

def girl_by_id(girl_id):
    for girl in girls:
        if girl.id == girl_id:
            return girl
    return None
```

```
@login_manager.request_loader
def request_loader(request):
    token = request.headers.get('Authorization')
    token = token.replace('Basic', '', 1) if token else ''
    id = decode_token(token)
    return girl_by_id(id)

def decode_token(token):
    return base64.b64decode(token).decode("utf-8")

@app.route('/girls/<string:id>', methods = ['GET'])
@flask_login.login_required
def get_girl(id):
    json_girl = mixin_to_json(girl_by_id(id))
    return jsonify(json_girl)
```

Figura 20. Representação de um pedaço de código de autenticação

A parte da gestão de sessão diz respeito a como se dará a troca de session id entre o cliente e o servidor. O risco aqui explicito é a interceção da identidade de uma pessoa logada permitindo a pessoas não autorizadas realizarem as mesmas atividades de uma pessoa autorizada.

4.1.5 WEB PAGE

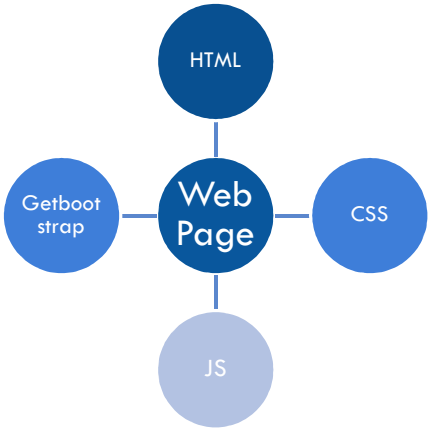
4.1.5.1 Análise de requisitos

Após termos a nossa API criada, o ultimo passo para que o nosso projeto esteja realizado seria mostrar que está tudo a funcionar corretamente de uma forma mais dinâmica, e para isso criamos uma Web Page. A principal razão de a termos criada, foi para darmos mais dinamismo ao trabalho, pois apenas ver pedidos realizados à API pela ferramenta Postman não tinha tanto impacto.

Essa Web Page foi criada usando HTML, CSS e JavaScript. Utilizamos também alguns templates oriundos do Getbootstrap.

A Web Page utiliza bibliotecas online para manter o aspeto que se pretende, e sem internet essas bibliotecas não podem ser utilizadas. Para evitar esse problema fizemos o import para que se possa visualizar a nossa Web Page da maneira pretendida com ou sem internet dando assim mais flexibilidade à mesma.

Em suma, de forma resumida, as tecnologias usadas foram:



4.1.5.2 Descrição e enumeração das tecnologias utilizadas

Para a criação da nossa Web Page utilizamos três linguagem principais para a criação da mesma, e por isso antes de explicarmos como as utilizamos na nossa Web Page, vamos fazer uma breve introdução sobre as mesmas:

- JavaScript (JS): esta é uma linguagem de scripting orientada a objetos desenvolvida pela Netscape e que é usada em milhões de páginas em todo o mundo. É uma linguagem bastante "leve" sem necessitar de compilação e é embebida em páginas HTML (cliente-side programming). O sucesso do JavaScript deve-se ao facto de, por exemplo, possibilitar a reação a eventos (o utilizador, ao carregar num botão, despoleta uma determinada ação) e de possibilitar a validação de informação antes de ser enviada para o servidor. O JavaScript não tem qualquer relação com o Java da Sun, no entanto tem uma sintaxe muito semelhante à do Java e à do próprio C++.
- HTML: é um acrónimo para a expressão inglesa *Hypertext Markup Language*, é a linguagem de publicação mais utilizada na web. Este formato é de utilização livre e baseia-se no SGML (*Standard Generalized Markup Language*). O HTML utiliza tags para estruturar e formatar o texto.
- CSS: é uma linguagem de estilos utilizada para definir a apresentação de documentos HTML. Por exemplo, o CSS controla fontes, cores, margens, linhas, alturas, larguras, imagens de fundo, etc. O consórcio W3C tem promovido bastante o uso de CSS desde o ano em que foi fundado, em 1994, e já produziu diversas recomendações (CSS1 e CSS2). Vários browsers suportam CSS, entre eles encontram-se o Internet Explorer, o Firefox e o Opera. A principal vantagem desta linguagem é a separação entre a formatação e o conteúdo do documento.

Com as três tecnologias referidas anteriormente, criamos a nossa Web Page para o projeto ImoPrediction.

Uma Web Page é um documento que pode ser exibido num navegador da Web, como Firefox, Google Chrome, Edge, sendo assim definido como uma página web. As páginas da Web típicas fornecem hipertexto que inclui uma barra de navegação ou um menu da barra lateral com links para outras páginas da Web por meio de hiperlinks. Como exemplo do que referimos, podemos observar os seguintes campos da nossa Web Page:

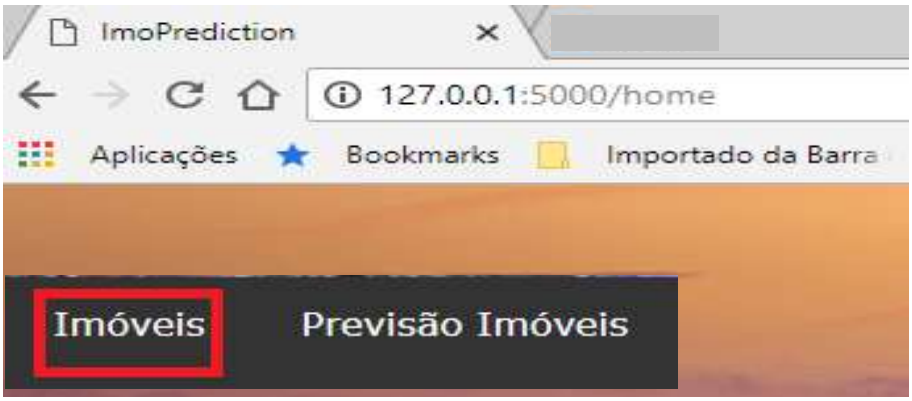


Figura 21. Representação das hiperligações do nosso site

Na imagem a cima podemos observar parte do site. Se carregarmos na secção Imóveis vamos abrir outra Web Page, que no nosso caso vai ser a /escolha como podemos ver aqui:

Figura 22 *Exemplo Web Page*

Como referimos anteriormente para podermos criar o nosso Web Page tivemos de utilizar três linguagens de programação. A forma como estas linguagens foram aplicadas são:

- HTML: foi utlizado para estruturar a Web Page, nele criamos tabelas para o utilizador poder visualizar melhor os dados.
- CSS: foi utilizado para fazer o design das páginas, com ele podemos criar, manipular a Web Page de maneira a ficar mais estética. No nosso caso além de usarmos para ter uma melhor apresentação também foi usado para inserir imagens.
- JavaScript: foi utilizado para criar scripts que dão as funcionalidades às páginas. Cada hiperligação, ou cada botão, a sua funcionalidade foi criada tendo por base esta linguagem.

Como se trata de um trabalho Web e têm que ter componente visual, para o utilizador visualizar os dados, tentamos dar o nosso melhor para criarmos tabelas onde iam ser impressas os dados. A ideia era aparecer imagens das casas e as suas descrições para baixo, mas devido ao facto de nunca termos trabalho com a parte da web, a melhor coisa que conseguimos fazer foi criar, por exemplo, a seguinte tabela:

Numero de Resultados: 282

Localização	Tipo de imovel	Tipologia	Condição	Nº quartos	Nº WC	Area Util	Preço
Loures, Sacavém e Prior Velho	Apartamento	T2	Usado	4	2	60	106000
Lisboa, Santo António	Apartamento	T6	Usado	4	3	134	698000
Loures, Loures	Apartamento	T4	Novo	3	1	0	395000
Cascais, Cascais e Estoril	Apartamento	T3	Usado	1	1	0	550000
Mafra, Ericeira	Apartamento	T3	Emconstrução	2	2	177	360000
Lisboa, Campo de Ourique	Apartamento	T2	Porrecuperar	2	2	68	385000
Loures, Sacavém e Prior Velho	Apartamento	T3	Usado	4	3	145	390000

Figura 23. Representação de alguns dados no formato da Web Page

A figura acima representa uma tabela da nossa Web Page, esta tabela é o resultado de uma pesquisa por casas para comprar.

Além de facilitar a visualização dos dados, o HTML foi também usado para permitir ao utilizador inserir as caraterísticas que pretende. Na imagem a baixo podemos ver um exemplo:

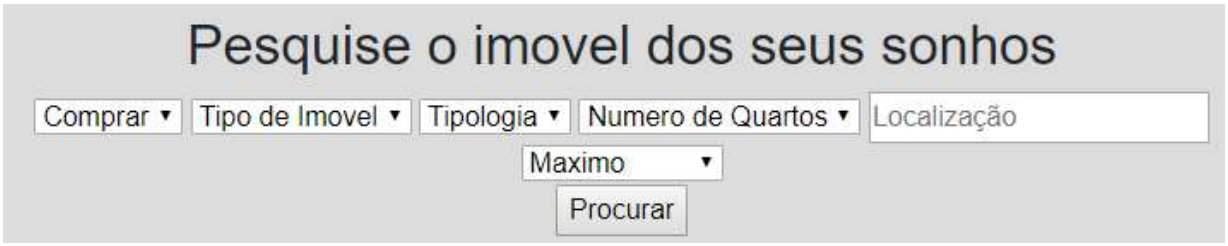


Figura 24. Motor de busca da nossa Web Page

As caraterísticas que o utilizador deseja inserir são tratadas da seguinte maneira: em primeiro lugar o utilizar vai escolher através de uma lista de opções e/ou escrever as caraterísticas que pretende que a casa tenha. De seguida a Web Page vai receber esses inputs e vai enviar para a API. A API vai receber os inputs enviados da Web Page e vai inseri-los numa query e vai pedir à base de dados a informação contida naquela query. Por sua vez a base de dados, vai analisar a query e vai enviar a informação pretendida. Após isso a API devolve a informação à Web Page que vai afixar essa informação para o utilizador ver.

O JavaScript foi utilizado para fazer scripts para a Web Page, como já tinha sido mencionado. Uma das funcionalidades que utilizamos foi para esconder caixas de inputs com o objetivo de só serem visíveis após certas condições. As seguintes figuras ilustração isso:

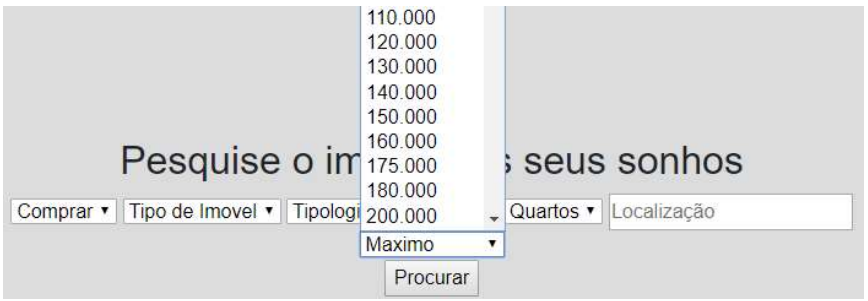


Figura 25 Resultado do motor de busca-Comprar

Na figura acima o utilizador escolheu a opção comprar e essa escolha foi ativar a lista de preços para venda, escondendo os preços de arrendar.

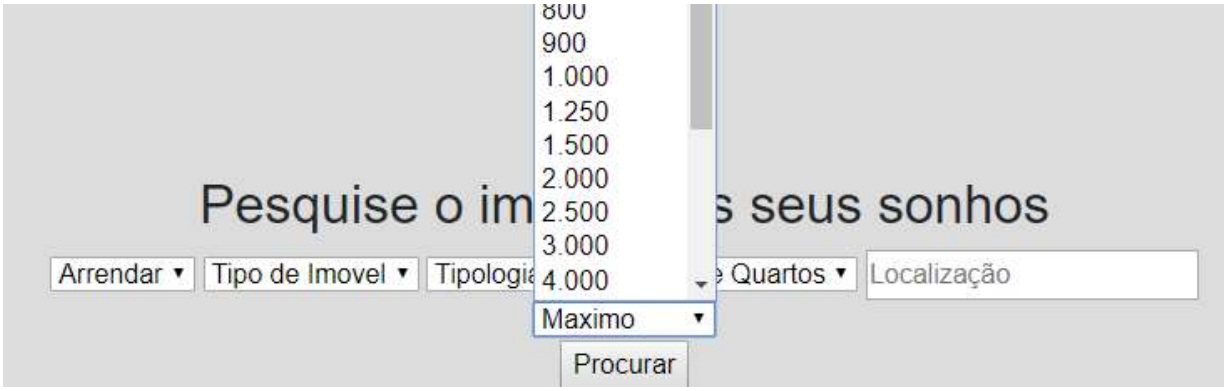


Figura 26 Resultado do motor de busca-Arrendar

Na figura acima o utilizador escolheu a opção arrendar e tal como na figura anterior, essa escolha foi ativar a lista de preços, mas desta vez foi ativar a lista de preços para arrendar escondendo os preços de comprar. Nós escolhemos fazer essa separação de preços porque achamos que seria mais sensato pois não existem em Lisboa casas a vendar por 800 euros nem casas para alugar por 10000 euros. Tal foi conseguido através de uma função criada em JavaScript que funcionava da seguinte forma: a função (TogglePrices) utiliza um simples if que ia verificar que valor estava numa variável chamada Tipo_Negocio, essa variável é que define se o utilizador pretende arrendar ou comprar uma casa. Quando o valor do input da variável Tipo_Negocio era comprar a função ToggePrices ia adicionar uma função chamada hide á lista arrendar e vai remover essa mesma função á lista comprar. Quando o valor do input era arrendar dava-se o inverso.

```
function TogglePrices(select) {
    console.log(select.value);

    if (select.value == "venda") {
        document.getElementById("Arrendar").classList.add('hide');
        document.getElementById("venda").classList.remove('hide');
    } else if (select.value == "Arrendar") {
        document.getElementById("venda").classList.add('hide');
        document.getElementById("Arrendar").classList.remove('hide');
    }
}
```

Figura 27. Pedaço de código que mostra como o hide foi efetuado

Temos também uma submenu dentro do campo de imoveis, onde o utilizador pode visualizar as casas pelas categorias que desejar. O submenu que se encontra dentro da página de imoveis, irá abrir outras páginas conforme o que o utilizador deseja procurar. Iremos então colocar aqui um exemplo de uma das páginas que o utilizador pode, caso entre nesse submenu:

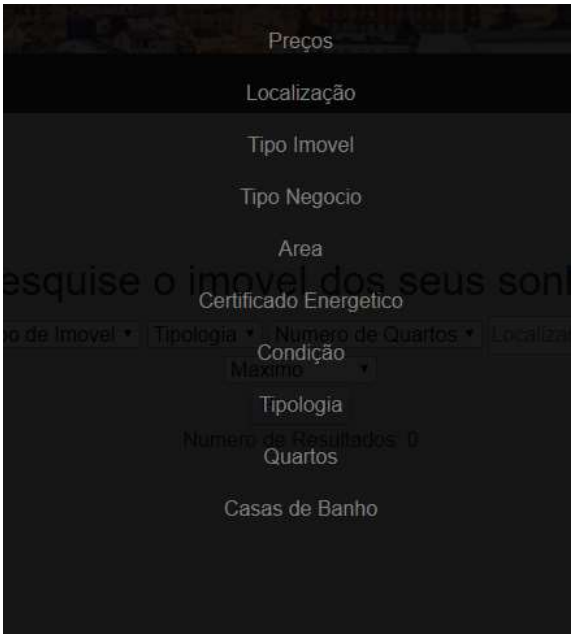


Figura 28. Representação do submenu dos imoveis

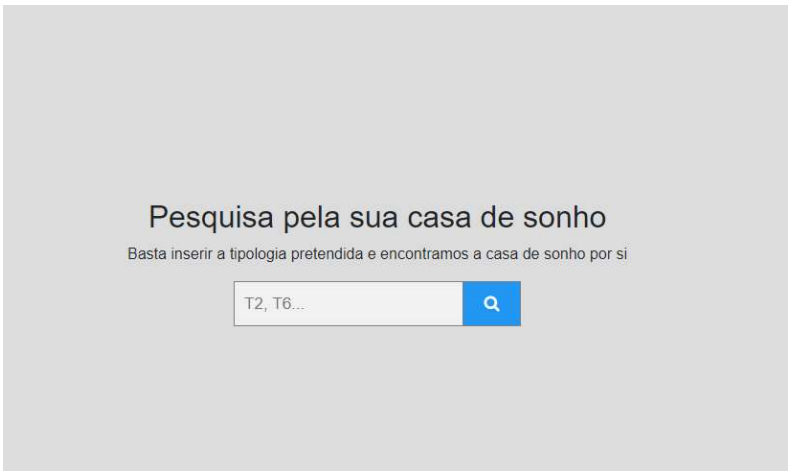


Figura 29. Representação de uma tela do submenu

Todas as partes aqui mencionadas da Web Page eram relativas à parte da pesquisa de imobiliários.

Na parte relativa às previsões, não as temos por categorias, temos apenas um motor de busca onde o utilizador coloca as características que pretende que a sua casa tenha e depois é-lhe então impresso na mesma página o preço. Através da seguinte imagem podemos comprovar isso:

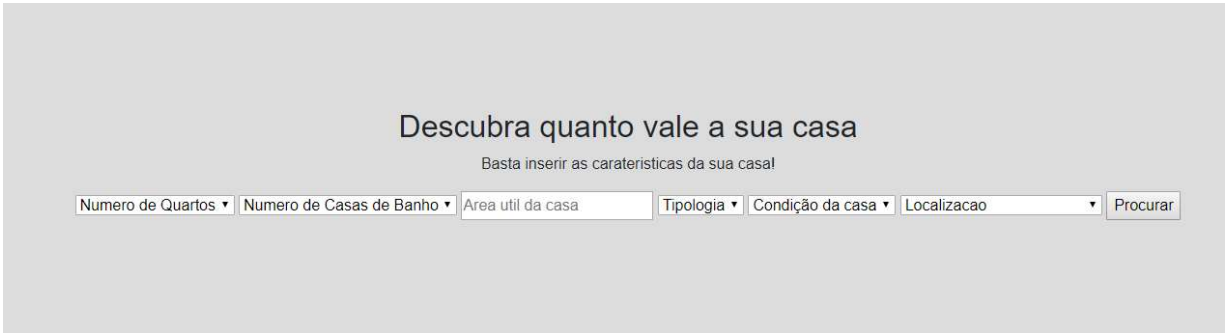


Figura 30. Web Page de previsão para venda de casas.

Capítulo 5

5.1 Conclusão

Durante o desenvolvimento deste projeto encontramos inúmeras dificuldades e obstáculos desde o primeiro ponto do projeto até ao último.
Na parte dos Web Crawlers, tivemos bastante dificuldade a retirar toda a informação que queríamos, uma vez que cada site tem a sua maneira de organizar os dados e por isso o que num dava, no outro já não funcionava. Além disso nem todas as caraterísticas que uma casa de um site continha existiam no outro, sendo outro obstáculo, que também acabava por dificultar quando fossemos fazer os métodos pois ficávamos com muitos valores nulos o que iria levar a uma precisão menos correta do nosso modelo. Para resolvermos esse problema decidimos diminuir as caraterísticas das casas escolhendo só aquelas que apareciam em ambos os sites, contudo mesmo assim havia um ou outro anúncio que não tinha uma determinada caraterística pelo que levou há existência de valores nulos, mas desta vez em menos quantidade. Além disso, os dados retirados não tinham os valores do preço representados da mesma maneira, num o preço era representado com um espaço no meio e o outro com uma virgula no meio, pelo que este problema foi superado quando normalizamos os dados.

Em relação dos métodos, resolvemos utilizar, como já referido, a regressão linear e o Random Forest. Na nossa primeira tentativa a previsão estava muito má e para conseguirmos melhora-la dividimos os dados em: dados de venda e dados de aluguer. Com essa divisão já nos foi possível atingir uma previsão com um acerto de 91% para o arrendar e de 95% para o comprar. Já a regressão linear não conseguimos ter um bom resultado e foi usado apenas para fazer comparação entre os dois métodos. Encontramos bastantes dificuldades para conseguirmos inserir o nosso método na API porque para podermos usar o método tivemos de transformar as strings tal como falamos na secção do método. Contudo quando fomos inserir na API e na Web Page tivemos bastantes dificuldades em saber como o iriamos fazer. Felizmente conseguimos implementar através de uma

lista que transforma outra lista e assim conseguimos implementar os métodos com sucesso. De realçar que este projeto não terminou como queríamos, pois, existem pequenos pormenores que não conseguimos implementar e que iriam melhorar bastante este projeto tal como um método de criptografia para as contas de autentificação.

A nosso ver a principal razão de não termos conseguido implementar esses pequenos detalhes foi devido à falta de tempo, visto que as únicas bases técnicas que tínhamos era sobre base de dados e o Machine Learning, pelo que tudo o resto requisitou um esforço maior em termos de pesquisa para conseguirmos perceber como funciona e como implementar.

Felizmente conseguimos realizar o que foi pedido, além disso conseguimos aprofundar o nosso conhecimento o que nos agradou imenso. Apesar de terminado, gostaríamos de continuar, no futuro, a melhorar este projeto pois gostamos imenso do tema e queremos aperfeiçoar o máximo possível, pois achamos que ao fazê-lo iremos conseguir aumentar os nossos conhecimentos e iremos construir um excelente projeto que muitos imobiliárias poderiam gostar, visto que é algo inovador que estas poderiam implementar.

Capítulo 6

6.1 Apêndice

[1] Wikipédia - https://pt.wikipedia.org/wiki/Rastreador_web

[2] Globalad - http://globalad.com.br/blog/o-que-e-crawler/

[3] Wikipedia - https://en.wikipedia.org/wiki/Web_scraping

[4] Wikipédia - https://pt.wikipedia.org/wiki/Coleta_de_dados_web

[5] Scrapy] - https://scrapy.org

[6] Wkipédia - https://en.wikipedia.org/wiki/Scrapy

[7] Towards Data Science - https://towardsdatascience.com/a-flask-api-for-serving-scikit-learn-models-c8bcdaa41daa

[8] Wikipédia - https://en.wikipedia.org/wiki/Windows_Task_Scheduler

[9] AWS Amazon - https://aws.amazon.com/pt/s3/

[10] Wikipédia - https://en.wikipedia.org/wiki/Amazon_S3

[11] Full Stack Python - https://www.fullstackpython.com/celery.html

[12] Wikipédia - https://en.wikipedia.org/wiki/MongoDB

[13] Towards Data Science - https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

[15] Machine Learning Mastery - https://machinelearningmastery.com/linear-regression-for-machine-learning/

[16] Data Blogger - https://www.data-blogger.com/2017/11/29/house-price-prediction-using-random-forest-classifier/

[17] Medium- https://medium.freecodecamp.org/what-is-an-api-in-english-please-b880a3214a82

[18] Canal Tech - https://canaltech.com.br/software/o-que-e-api/

[19] Becode - https://becode.com.br/o-que-e-api-rest-e-restful/

[20] Medium - https://medium.com/python-pandemonium/build-simple-restful-api-with-python-and-flask-part-1-fae9ff66a706

[21] Towards Data Science - https://towardsdatascience.com/a-flask-api-for-serving-scikit-learn-models-c8bcdaa41daa

[22] AWS Amazon - https://docs.aws.amazon.com/pt_br/athena/latest/ug/parsing-JSON.html

[23] Celery Project - http://docs.celeryproject.org/en/latest/getting-started/first-steps-with-celery.html

[24] Redis - https://redis.io/

[25] Medium - https://medium.com/@roselmamendes/python-construindo-aplica%C3%A7%C3%B5es-web-seguras-com-flask-fe8989e282ac

Anexos

A. Manual de utilizador

1º Passo:

Instalar e iniciar uma aplicação, exemplo wampserver64, que consiga fazer de servidor para o MySQL.

2º Passo (opcional):

Com o servidor ligado correr os Crawlers através de uma linha de comandos como por exemplo a do anaconda. Nessa linha de comandos escrever o seguinte código: scrapy crawl (nome da pasta sem parenteses). Nota: as pastas onde se encontra os Crawlers contêm o nome necessário. Este passo só é necessário se pretender atualizar os dados da base de dados manualmente.

3º Passo:

ir a uma linha de comandos que corra python. Através da linha de comandos ir a pasta onde se encontra a api.py.

4º Passo:

Correr na linha de comandos o seguinte código: python api.py, e esperar que o servidor arranque

5º Passo:

No browser ir para este endereço: http://127.0.0.1:5000

6º Passo:

Carregar na opção iniciar a Sessão e escolher a opção efetuar registo, caso não tenha ainda conta. Se já tiver conta apenas insira os seus dados e carregue no entrar.

7º Passo:

Existem três opções que pode escolher. Pode ir a procura de casas para comprar ou alugar clicando em **Imoveis**. Pode descobrir quando vale a sua casa caso queira vender ou alugar carregando na opção **Previsão imóveis**. E por fim pode carregar na opção **sair** para sair da sua conta.

8º Passo:

Quando terminar de ver tudo o que queria basta desligar o servidor da api.

Passos extra:

Dentro da opção **Imóveis** tem duas opções meter apenas as caraterísticas que pretende ou, ir a opção **Pesquisar categorias** e pesquisar por apenas uma caraterística que lhe interesse.

Dentro da opção **Previsão Imóveis** tem duas opções **Arrendar** ou **Comprar** se carregar no arrendar vai poder saber quanto pode por a sua casa a arrendar, se escolher comprar vai poder descobri quando vale a sua casa para poder vender.