Nanodegree Engenheiro de Machine Learning

Projeto final: Prevendo salários de vagas de emprego

Leonardo Mizoguti – 30 de Abril de 2018

Definição

Visão geral do projeto

No mercado de recrutamento, existem muitos fatores a serem considerados para se fazer o *matching* entre candidatos e oportunidades de emprego. Um dos mais importantes é a **conformidade do salário** proposto mediante as **expectativas salariais** dos candidatos.

Muitas vezes empresas preferem **não divulgar o salário** para uma determinada vaga, portanto um sistema de *matching* deveria ser capaz de **estimar quanto seria o salário proposto** para poder atrair e recomendar os candidatos com uma **pretensão salarial correspondente**.

A proposta desse projeto é desenvolver um modelo capaz de **fazer uma estimativa de salário** baseando-se nos dados de vagas (título, descrição, localidade, etc.) provenientes de um conjunto de dados de vagas baseadas nos Estados Unidos postadas no site de empregos *Monster.com* [1].

Descrição do problema

O objetivo do projeto consiste em desenvolver um **modelo de regressão** que recebe como entrada um conjunto de **dados categóricos** sobre determinada vaga (como localização, tipo, setor…) bem como **seu título** e **sua descrição** (ambos campos de texto livre, não estruturados), e retorna um **valor numérico** correspondendo a uma estimativa do **salário proposto** para a vaga.

Os dados deverão ser **pré-processados** para que alguns modelos de regressão possam ser **treinados** e em seguida **comparados**. Por fim, um dos modelos deverá ser **escolhido** e **ajustado**, a fim de se obter o resultado final.

Métricas

A performance dos modelos treinados deverá ser medida usando-se o **desvio absoluto médio (DAM)**. Basicamente essa métrica mede, na média, o quanto o **valor previsto** pelo modelo **diverge** (em termos absolutos) do **valor proveniente dos dados**.

$$\text{Erro} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Onde n é o número de vagas, y_i corresponde a um valor de salário previsto pelo modelo para uma vaga e x_i ao valor real do salário da mesma vaga.

Essa é uma boa métrica pois pode-se ter uma noção do tamanho do erro na média com relação à média do valor dos salários.

Análise

Exploração dos dados

O conjunto de dados utilizados para a concepção do modelo está publicamente disponível na plataforma *Kaggle* [2] e consiste de 22.000 vagas de emprego postadas no site de empregos *Monster.com*.

Em uma primeira análise do conjunto, percebeu-se que esse conjunto possui algumas colunas que não são relevantes para a análise e portanto decidiu-se não considerá-las. São elas:

country	Sabe-se que todas as vagas são baseadas nos Estados Unidos
country_code	Mesma lógica da coluna *country*
date_added	Esse campo não está preenchido para a grande maioria das vagas
has_expired	Nenhuma das vagas foi marcada como expirada
job_board	Todas as vagas foram postadas no site *Monster.com*
page_url	A URL da vaga não representa informação útil para o salário
uniq_id	O identificador da vaga não representa informação útil para o salário

Dessa forma, o conjunto utilizado apresenta 7 colunas úteis para o modelo. São elas:

job_description	Descrição da vaga
job_title	Título da vaga
job_type	Tipo da vaga (tempo integral, meio período, temporário, etc)
location	Cidade e estado nos quais a vaga está baseado
organization	Segmento da economia da empresa
sector	Tipo de vaga (gerência, vendas, engenharia, etc)
salary	Salário

Seguem alguns exemplos de vagas:

Japanese and English Speaking Interpreter and Translator for Toyota Motor Mgf. Job in Charleston

Job type: Full Time
Location: Charleston, WV
Organization: Manufacturing - Other
Sector: Experienced (Non-Manager)

***Salary:* 58,000.00 - 65,000.00 $ /year**

Job Description: Technical Japanese/English Speaking Interpreter/Translator for Toyota Motor Manufacturing To Apply: Send resumes to tre@ictnorthamerica.comJob Dates: May 30, 2016 to November 25, 2016 Location: West VirginiaJob Openings: ICT is looking for Japanese/English interpreters/translators to assist as written and spoken communication bridge between American and Japanese engineers. These individuals will be working on-site at the Toyota Motor's plant. Interpreting and translation will take place in business and engineering meetings, on the plant floors, and in the plant office.Requirements: Must be able to speak both English and Japanese fluentlyMust be able to read and write Japanese kanji at a fluent level (Note: our scale for kanji fluency is based on being able to read 75% or more of a standard Japanese newspaper) Must have good interpersonal skills Must have previous interpreting and translation experience Must be willing to work onsite at the Toyota facility Four year degree or equivalent experience Good verbal and written communication skillsA good team member attitude Benefits: Salary is based on the individual's experienceForty plus hours per week Overtime paid at time and a half Housing provided Meal allowance Insurance after 3 monthsTravel Paid Please contact us if you would like more information about working on ICT's team. For more information or to apply through ICT,LLC see www.ictnorthamerica.com

Senior Accountant/Analyst Job in Denver

Job type: Full Time Employee
Location: Denver, CO 80202
Organization: <null>
Sector: Accounting/Finance/Insurance

***Salary:* 70,000.00 - 85,000.00 $ /year**

Would you like to grow your accounting and finance career with a great start-up type environment with an un-matchable culture? We have an exciting job opportunity in Denver, CO for a hybrid senior accountant/analyst. You will have the opportunity to learn from someone with extensive public accounting and start-up experience who is also a pleasure to work with. You will be responsible for all accounting functions as well as implementing policy and procedures and framing the accounting department. In addition, you will play a strong role with relaying financial information with the Director of Finance and Board of Directors. To be considered for this position you must hold a Bachelor's in Accounting or Finance and have at least two years in public accounting. Great opportunity for relocation! Please email your resume directly to matthew.diggins@parkerlynch.com As a senior accountant/analyst your responsibilities will include: Monthly accounting close:Monthly close analysis, journal entries and account reconciliations for areas including, accrued liabilities, deferred revenue, prepaid expenses, fixed assets, product cost, inventory, intercompany, debt, cashFP&A — Budgets, Forecasts, (expenses and revenues), weekly cash flow projections, balance sheet estimatesAccount variances and financial statement analysis comparisons to budget, forecasts and prior yearAssist in monthly, quarterly and annual reporting packages and financial statementsAssist with the annual audit process Your qualifications: Bachelor's degree in Accounting or FinanceAt least two years in public accounting preferably with a national firmCPA preferredStrong skills in MS office If you have a passion for growing your career and skill set we would love for you to be a part of our client's team in Denver, CO.

Com uma análise mais detalhada dos valores presentes, chegou-se a conclusão que um pré-processamento seria necessário. Seguem as análises feitas para cada atributo.

Salário (salary)

O salário é o atributo central do modelo, dado que é a **variável alvo da regressão**. Dito isso, simplesmente não é possível utilizar as vagas que não possuem uma indicação de salário. Do total de 22.000 vagas, somente 3299 possuem o campo *salary* não nulo.

Além disso, não basta o campo *salary* ser não-nulo, seu conteúdo deve **objetivamente indicar um valor numérico**, caso contrário ele não é útil. No conjunto de dados, existem indicações como as mostradas abaixo, que não são úteis.

- *"Excellent benefits package, Health, 401k"*
- *"Negotiable"*
- *"Competitive base salary, commission and a comprehensive benefit package"*

Se forem consideradas somente as vagas cujo campo *salary* possui um valor padronizado de salário – da forma `[\d,]+\.\d{2}` (e.g. "12.00", "95,315.43", …) – o número cai ainda mais para 2904.

Um primeiro pré-processamento necessário é a **remoção das vagas sem indicação objetiva de salário** e **a padronização de valores não padronizados**.

Além disso, a grande maioria das vagas com uma indicação objetiva de salário apresenta **um intervalo,** e não um único valor, como por exemplo:

- `"11.00 - 13.00 $ /hour"`
- `"80,000.00 - 95,000.00 $ /year"`
- `"1.00 - 1,000,000.00 $ /year"`
- `"0.00 - 150,000.00 $ /yearbonuses"`

Dos exemplos acima, é possível identificar três problemas. O primeiro é a existência de **valores aberrantes** como "1.00 - 1,000,000.00 $ /year", sendo bem claro que essa indicação é uma forma de simplesmente não informar o salário real.

O segundo é que os salários estão em **bases de tempo diferente**, i.e. são encontrados salários anuais, mensais, semanais, por hora, etc.

O terceiro é o problema do **intervalo**, é necessário estabelecer uma lógica de conversão de um intervalo para um único valor. No caso de "11.00 - 13.00 $ /hour", a média dos dois valores é um valor razoável, enquanto que no caso de "0.00 - 150,000.00 $ /yearbonuses", talvez seja melhor considerar o valor "150,000.00".

É necessário portanto **remover valores aberrantes**, **converter intervalos de valores em um único valor** e **normalizar os valores para uma mesma base de tempo**, caso contrário o modelo terá dificuldades de estimar salários com ordens de grandeza tão distintas**.**

Uma última consideração a se fazer é que em muitas vagas pode-se encontrar a indicação de salário no campo *job_description* como parte da descrição da vaga. Para aumentar o volume de dados, é necessário **extrair a indicação de salário da descrição da vaga** para as vagas que possuem o campo *salary* vazio.

Tipo de vaga (job_type)

O campo *job_type* é quase padronizado. Analisando a frequência dos valores obtidos entre as vagas com informações úteis de salário, obtém-se a seguinte relação:

job_type	Freq.	job_type	Freq.
Full Time	1766	Per Diem	5
Full Time Employee	1532	Job Type Full Time Temporary/Contract/Project	2
Full Time, Employee	337	Full Time , Temporary/Contract/Project	2
NaN	325	Part Time Temporary/Contract/Project	2
Full Time Temporary/Contract/Project	319	Part Time , Employee	2

Part Time	117	Full Time / Employee	2
Full Time , Employee	115	Part Time, Temporary/Contract/Project	2
Full Time, Temporary/Contract/Project	60	Per Diem, Employee	1
Temporary/Contract/Project	52	Job Type Part Time Employee	1
Employee	29	Part Time Intern	1
Part Time Employee	17	Full Time Employee	1
Part Time, Employee	10	Job Type Employee	1
Job Type Full Time Employee	7	Part Time , Temporary/Contract/Project	1

Pode-se perceber que alguns valores são os mesmos, com algumas vírgulas e outros caracteres sobressalentes. Será necessário **normalizar e agrupar esses valores**.

Localidade (location)

Segue abaixo alguns exemplos de valores encontrados nesse campo:

Phone 8658241190 Address KnoxvilleKnoxville, TN 37929
Columbus, OH 43228
Springfield, IL 62702
Clarks, LA 71415
Austin, TX
Columbus, OH
Part Time Deputy Judicial Marshal - Belfast

Percebemos alguns **valores problemáticos**, uma **falta de padronização** e uma **variabilidade grande de valores**. Será necessário tratar esses problemas.

Organização (organization) e setor (sector)

Esses dois campos são semelhantes, embora indiquem informações distintas. A grande maioria das vagas **não possuem esses campos preenchidos**, e os valores presentes **não são padronizados** e alguns apresentam problemas. Alguns exemplos:

Organization	**Sector**
AllInsuranceOther/Not Classified	Food Services/Hospitality
Automotive Sales and Repair Services	General/Other: Food Services

RetailBankingFinancial Services	Career Level Experienced (Non-Manager)
Columbus, OH 43016	Experienced (Non-Manager)

Será necessário tratar esses problemas, como uma indicação de localidade, setores repetidos mas indicados de forma diferente, concatenação de valores, etc.

Título da vaga (job_title)

Seguem alguns exemplos de títulos de vagas [sic]:

- Location Manager/ Funeral Director Job in Hilliard
- Quality Assurance Manager Job in Chicago
- Sales Representative (Advertising & Marketing) Job in Anniston
- Chemical Operations Specialist Job in Omaha body { margin:px; overflow: visible !important; } #ejb_header { color: #; font-family: Verdana
- Part Time Deputy Judicial Marshal - Belfast Job in Belfast
- PROJECT MANAGER - SCHEDULING Job in Irvine

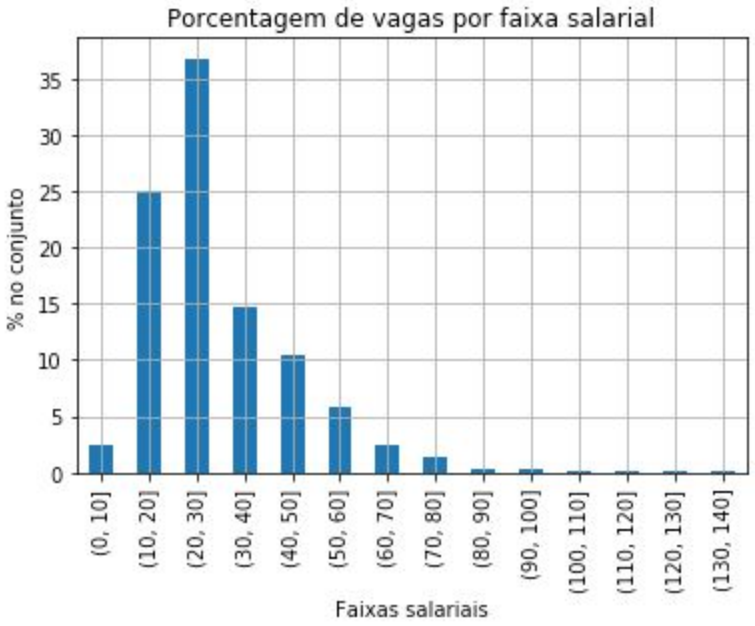
Existem alguns **padrões** nos títulos das vagas que **poderiam ser removidos** para melhorar a análise, como por exemplo a indicação da localidade ao final de cada título. Existem alguns outros elementos que devem ser removidos, como por exemplo o caso do código CSS presente no título (certamente um problema da extração).

Descrição da vaga (job_description)

Da mesma forma como no título, existem **indicações** nas descrições que podem ser **removidas** ou tratadas para melhorar a acurácia da análise, como na primeira vaga exemplo, vê-se indicação de e-mail e do site da empresa, informações não relevantes para a descrição na estimativa de salário.

Visualização exploratória

Após o pré-processamento de salários, pode-se analisar a porcentagem que contém uma determinada faixa de valores. Com quebras em faixas de $10, obtém-se o seguinte gráfico:



Observa-se que a maioria das vagas possui salário entre $10 e $30 por hora. Dado esses dados, é razoável esperar que um bom modelo forneça um erro absoluto médio **abaixo de $10 por hora**.

É interessante remover valores muito altos, pois é provável que o erro dessas vagas irá **distorcer a métrica** de qualidade do modelo. É de se esperar que o erro entre essas vagas será maior tanto em termos relativos (pois há menos vagas nessas faixas) como em termos absolutos.

Algoritmos e técnicas

Representação de informação textual não estruturada

Para a transformação dos atributos textuais das vagas em vetores numéricos, a proposta é utilizar e comparar duas técnicas distintas: *TF-IDF com bi-grams* e *Word2Vec*.

Resumidamente, a técnica *TF-IDF* [3] (term frequency - inverse document frequency) *com bi-grams* analisa a frequência (e por consequência importância e relevância) de palavras e pares de palavras (bigrams) dentro de cada documento (título e descrição de uma vaga) e no contexto de todos os documentos analisados (todos os títulos e descrições de todas as vagas), gerando para cada documento um vetor numérico que o representa dentro de um espaço vetorial construído de tal forma que cada *unigram* e *bigram* representa uma dimensão.

Já a técnica *Word2Vec* [4] utiliza uma rede neural treinada para analisar o contexto em que palavras são usadas e representar palavras semanticamente próximas como vetores próximos dentro de um espaço vetorial com um número limitado de dimensões.

O Google [5] fornece publicamente um modelo pré-treinado com 100 bilhões de palavras e com vetores de 3 milhões de palavras e frases em um espaço vetorial de dimensão 300.

Modelo regressor

Foram escolhidos 3 métodos de regressão diferentes para comparação: máquinas de vetores de suporte (SVR), *k-nearest neighbors* (KNN) e *random forest*.

As máquinas de vetores de suporte é um bom método candidato para o problema, pois devido à transformação textual em vetores numéricos, o problema de regressão deve ser resolvido em um espaço vetorial com um alto número de dimensões, e as SVM em geral produzem bons resultados com tais espaços.

Podemos argumentar que, pelo domínio do problema, o método KNN também é um bom candidato, assumindo que vagas de emprego similares (com títulos similares, descrições similares, na mesma localidade, etc) possuem remunerações similares.

Por fim, o método random forest foi escolhido pois trata-se do método que melhor performou no modelo de benchmark apresentado na próxima seção. Random forests em geral performam muito bem em uma ampla variedade de problemas.

Benchmark

Há aproximadamente 5 anos, a plataforma *Adzuna* promoveu na plataforma *Kaggle* uma competição [6] cujo objetivo era também **prever salários** de vagas de emprego baseadas no Reino Unido com base em um conjunto de atributos.

Jackman, S. e Reid, G., da University of British Columbia, publicaram [7] no mesmo ano um artigo no qual eles compararam diferentes modelos na resolução do problema proposto pela *Adzuna*, a saber, **regressão LASSO**, **regressão por máxima verossimilhança**, **regressão por rede neural** e **regressão por *random forest***.

A conclusão da dupla foi de que o modelo de **random forest** foi o que melhor performou, com um desvio absoluto médio de **£5,000/ano** no conjunto de teste.

O resultado final do projeto poderá ser comparado, dado que será usada a mesma métrica. Será necessário comparar os dois resultados na mesma moeda e na mesma base de tempo.

Metodologia

Pré-processamento de dados

Será fornecida aqui uma versão resumida (porém completa) dos pré-processamentos feitos no conjunto de dados. O pré-processamento detalhado está documentado junto ao código fonte fornecido com o projeto.

Pré-processamento de *salary*

- Removeram-se indicações não úteis de salário (e.g. "Excellent benefits package, Health, 401k");
- Foram extraídas indicações existentes de salários padronizadas nas descrições (campo *job_description*) quando o campo salary estava vazio;
- Padronizaram-se indicações não padronizadas (e.g. O valor "$75/HR with Bonus" foi convertido para "75.00 hour")
- Adaptaram-se intervalos de salário conforme a convenção:
    - Quando ambos os limites são bem definidos (e.g. "10.00$ - 12.00$"), é usada a média dos dois valores;
    - Quando apenas um dos limites é bem definido (e.g. "0.00 - 50,000.00 $ /year" ou "$70,000+"), é usado o valor do limite bem definido.
- Todos os salários foram convertidos para a mesma base de tempo: **dólares por hora**. Foram consideradas a seguintes premissas:
    - Uma dia de trabalho consiste de 8 horas;
    - Um fim de semana de trabalho consiste de 16 horas de trabalho;
    - Uma semana consiste de 40 horas de trabalho;
    - Um mês consiste de 160 horas de trabalho;
    - Um ano consiste de 1920 horas de trabalho.
- Valores aberrantes foram removidos, sendo somente permitidos salários entre $7.25 (salário mínimo americano) e $70 por hora.

Pré-processamento de *job_type*

- Foram removidos caracteres e substrings sobressalentes e alguns valores foram agrupados;
- Valores que faltavam foram preenchidos com "Not classified".

Pré-processamento de *location*

- Para melhorar a análise dessa informação, foi extraída somente a indicação do estado americano no qual a vaga está baseada (sigla composta de duas letras maiúsculas);
- Vagas sem indicação de estado foram indicadas com o valor "Unknown".

Pré-processamento de *organization*

- Valores que faltavam foram preenchidos com "Other/Not classified";
- Indicações de localidade nesse campo foram substítuidas pelo valor "Other/Not classified";
- Valores indicando concatenação de vários valores diferentes foram substituídos pelo valor "Other/Not classified". Esses casos foram identificados pela presença de uma letra maiúscula seguindo uma letra minúscula (e.g. RetailBankingFinancial Services).

Pré-processamento de *sector*

- Valores que faltavam foram preenchidos com "Other";
- Algumas substrings sobressalentes (a saber, "Career level" e "General/Other:") foram removidas e alguns valores agrupados;
- Valores indicando concatenação de vários valores diferentes foram substituídos pelo valor "Other/Not classified". Esses casos foram identificados pela presença de uma letra maiúscula seguindo uma letra minúscula (e.g. Systems Analysis - ITWeb/UI/UX Design).

Pré-processamento de *job_title*

- Foram removidas indicações de localidade (e.g. Quality Assurance Manager **Job in Chicago**);
- Foram removidos códigos CSS;
- Foram removidos todos os caracteres que não representam letras do alfabeto inglês;
- Foram removidas palavras compostas de uma única letra;
- Foram removidas substrings não relevantes para o título (e.g. "per hour", "male", "female", "part time", etc);
- Foram removidos espaços sobressalentes;
- Os títulos foram convertidos em vetores numéricos conforme descrito na seção "Implementação".

Pré-processamento de *job_description*

- Foram removidas indicações de email e URLs;
- Foram removidos códigos CSS;
- Foram removidos todos os caracteres que não representam letras do alfabeto inglês;

- Foram removidas palavras compostas de uma única letra;
- Palavras concatenadas foram separadas (esses casos foram identificados por letras minúsculas seguidas de letras maiúsculas);
- Foram removidos espaços sobressalentes;
- As descrições foram convertidas em vetores numéricos conforme descrito na seção "Implementação".

Remoção de *outliers*

Conforme discutido na seção *Visualização exploratória*, é interessante remover vagas com salários muito altos para evitar distorções da métrica de qualidade do modelo. Dado que a vagas com salário acima de $70 por hora não chegam a representar 5% do total de vagas analisadas, elas foram desconsideradas.

Implementação

Carregamento e pré-processamento dos dados

Os dados foram carregados do CSV com os dados originais em um *DataFrame* do *pandas*. Todos os pré-processamentos descritos na seção anterior foram realizados com os recursos oferecidos pelo *pandas, numpy* e a biblioteca nativa de processamento de expressões regulares do Python, *re*.

Transformação de atributos categóricos

Os atributos categóricos (*job_type, location, sector* e *organization*) foram transformados em vetores (sparse matrix) com a ajuda da classe *DictVectorizer* do *scikit learn*.

Em seguida esses vetores foram agrupados aos vetores que representam os títulos e as descrições por meio da função *hstack* do *SciPy*.

Transformação de títulos e descrições em vetores numéricos

Para o método TF-IDF, os títulos e descrições foram transformados em vetores por meio da classe *TfidfVectorizer* do *scikit learn* ajustada para tokenizar os textos em unigrams e bigrams, removendo stop words da língua inglesa.

Para o método *Word2Vec*, o modelo pré-treinado do Google foi carregado usando a biblioteca *gensim*. Os textos foram tokenizados em unigrams, que em seguida foram buscados dentro do modelo. Por fim, o vetor final para cada título/descrição foi obtido calculando-se a média de todos os vetores não nulos.

Validação do modelo

A função *train_test_split* do *scikit learn* foi usada para separar conjuntos de treinamento (80% dos dados) e teste (20% dos dados).
A função *cross_val_score* foi usada em seguida para realizar a validação cruzada com 5 conjuntos diferentes para cada método regressor analisado. O score final de cada método foi calculado usando a média dos 5 valores obtidos na validação cruzada.

As classes *SVR*, *KNeighborsRegressor* e *RandomForestRegressor* do *scikit learn* foram usadas para treinar os modelos e obter os resultados.

Ajuste do modelo

O ajuste do modelo escolhido e o score final foram obtidos usando a classe *GridSearchCV* do *scikit learn*.

Refinamento

Na etapa de validação do modelo, os seguintes valores de erro absoluto médio foram encontrados para os modelos treinados:

Modelo	**TF-IDF**	**Word2Vec**
SVR	$9,82/hora	$9,73/hora
KNeighborsRegressor	$6,56/hora	$6,67/hora
RandomForestRegressor	$6,13/hora	$6,46/hora

Logo, verificou-se que o modelo a ser escolhido é o regressor Random Forest para ambos os conjuntos de dados, pois foi o que forneceu o menor erro absoluto médio por hora.

(OBS: dado que os dados de treinamento e teste são escolhidos aleatoriamente a cada execução do código, é possível que o modelo K-Nearest Neighbors forneça um erro ligeiramente menor do que RandomForest para o caso Word2Vec. Entretanto, na maioria dos casos testados, o método Random Forest forneceu melhores resultados).

A etapa de refinamento consistiu em ajustar os parâmetros *min_samples_split* e *max_features* do regressor por Random Forest usando o método de *grid search*.

O parâmetro *min_samples_split* representa o número mínimo de dados necessários em um nó para que em uma árvore de decisão este nó seja dividido em sub-árvores. O valor *default* deste parâmetro é 2. Além desse valor, testaram-se também os valores 10 e 50, que correspondem a 5 e 25 vezes o valor default respectivamente.

O parâmetro *max_features* representa o número máximo de atributos escolhidos aleatoriamente para a divisão de um nó em qualquer árvore de decisão pertencente à Random Forest. O valor *default* deste parâmetro é *n_features*, o que significa que todos os atributos são considerados. Os outros valores testados foram *sqrt* e *log2*, que correspondem à raiz quadrada do total de atributos e ao logaritmo em base 2 do total de atributos respectivamente. Reduzir o número máximo de atributos considerados aumenta o nível de aleatoriedade da floresta.

O número de estimadores do regressor foi configurado para 50 em todos os testes.

Ao final, os seguintes resultados foram obtidos:

	TF-IDF	**Word2Vec**
RandomForestRegressor (ajustado)	$5,90/hora	$5,86/hora

Resultados

Modelo de avaliação e validação

Para verificar a robustez do modelo, o treinamento e o teste foram executados 5 vezes com configurações diferentes de dados (a cada execução a função *train_test_split* com um *random_state* variável gera uma configuração diferente).

Os resultados não apresentaram grande variação. O modelo, sendo treinado, refinado e testado com configurações de dados diferentes apresentaram resultados consistentes.

Execução	**TF-IDF**	**Word2Vec**
1	$5,84/hora	$6,25/hora
2	$5,68/hora	$5,93/hora
3	$5,70/hora	$5,83/hora

4	$5,63/hora	$6,15/hora
5	$6,04/hora	$6,12/hora

Ao final, verificou-se que o modelo com melhor performance foi **Random forest**.

Na última execução do código, obtiveram-se os erros médios **$5,90/hora** para o conjunto **TF-IDF** e **$5,86/hora** para o conjunto **Word2Vec**.

Os parâmetros finais dos dois regressores foram os seguintes:

- **TF-IDF**

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=50, n_jobs=-1,
           oob_score=False, random_state=1, verbose=0, warm_start=False)
```

- **Word2Vec**

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=50, n_jobs=-1,
           oob_score=False, random_state=1, verbose=0, warm_start=False)
```

Justificativa

O modelo de benchmark forneceu como resultado final um erro absoluto médio de **£5.000** por ano (equivalente a **$7.103** por ano) em um modelo treinado com base em **244.768 vagas**.

O modelo apresentado neste relatório apresentou um resultado final de **$11.251** por ano (**$5,86** * 1920 horas por ano) em um modelo treinado com base em **3510 vagas**.

Em termos absolutos o modelo de benchmark forneceu um erro **36,6%** menor do que o modelo desenvolvido neste projeto, mas é importante considerar que este foi treinado com uma base de dados consideravelmente menor que aquele (1,4% do volume).

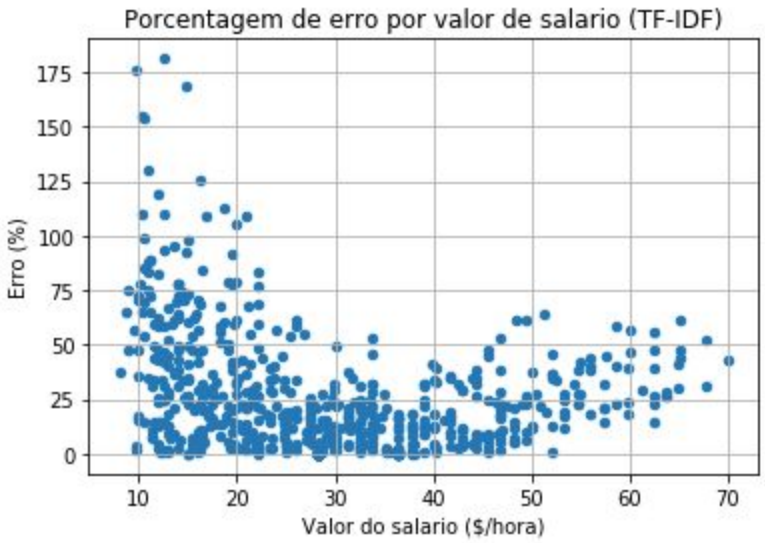
Mediante essa comparação, os resultados foram considerados satisfatórios, embora acredita-se que a performance do modelo poderia ter sido melhor.

Conclusão

Forma livre de visualização

Uma análise interessante é observar o tamanho do erro de acordo com o valor de salário da vaga. Pode-se plotar um gráfico no qual cada ponto corresponde a uma vaga, no eixo horizontal define-se o valor do salário a ser previsto e no eixo vertical a porcentagem de erro encontrada na previsão daquela vaga.

Podemos analisar o mesmo gráfico para os dois métodos de conversão de informação textual TF-IDF e Word2Vec:



Observa-se que para ambas as técnicas o erro percentual decresce com o aumento do valor do salário, atingindo os menores valores no intervalo de $30/hora a $40/hora, e em seguida o erro passa a aumentar novamente, provavelmente devido ao volume menor de vagas nesses faixas.

É interessante notar que o modelo que utilizou a técnica Word2Vec fornece erros percentuais menores para vagas com salários mais altos e erros maiores para salários mais baixos. O modelo TF-IDF foi melhor de maneira geral, mas o modelo Word2Vec performou melhor nas faixas salariais com maior densidade de dados.

Reflexão

A solução do problema apresentado por este projeto envolveu várias etapas. A primeira delas foi a análise do conjunto de dados, na qual identificaram-se os dados realmente úteis para a concepção do modelo e as necessidades de pré-processamento dos dados.

A segunda etapa consistiu em fazer o pré-processamento necessário, principalmente o tratamento dos salários, a variável alvo do modelo. Outro pré-processamento extremamente importante foi a transformação dos atributos textuais em vetores numéricos pelas técnicas TF-IDF e Word2Vec. Sem essa transformação, não teria sido possível utilizar essas informações no modelo de regressão.

A terceira etapa foi a validação do modelo. Três modelos de regressão foram treinados e comparados por meio de validação cruzada com os conjuntos de dados tratados com as duas técnicas de transformação de texto já mencionadas. Mediante os resultados, o modelo de regressão *RandomForest* foi escolhido para ambos os conjuntos de dados.

A quarta etapa consistiu no ajuste do modelo escolhido na etapa de validação. A técnica do *grid search* foi usada para comparar os resultados obtidos com os dados de treinamento variando-se o parâmetro *min_split_samples* do modelo *RandomForest*. Dessa forma obtiveram-se os resultados finais no conjunto TF-IDF e no conjunto Word2Vec.

Por fim a última etapa consistiu em fazer uma análise sobre a variação do erro percentual da predição mediante o valor real das vagas.

O maior desafio desse projeto foi o pré-processamento dos dados. A qualidade do conjunto de dados inicial não é muito boa no sentido de padronização. Foi necessário realizar várias etapas para extrair e padronizar salários, lidar com diversos valores problemáticos de atributos categóricos, além de configurar corretamente a transformação das informações textuais em vetores numéricos. Ambas as técnicas possuem tempos demorados de execução, mas em momentos diferentes: a técnica TF-IDF gera um espaço vetorial com um número alto de dimensões, o que torna o treinamento do modelo extremamente lento. Já o Word2Vec gera um espaço vetorial bem menor, mas o tempo de carregamento do modelo pré-treinado e da busca dos vetores nele contidos é considerável.

Apesar dos problemas, de maneira geral o modelo performou de maneira satisfatória segundo as expectativas iniciais. Talvez com um conjunto de dados melhor trabalho, ele possa fornecer resultados ainda melhores.

Melhorias

O modelo desenvolvido neste projeto poderia ser melhorado de diversas formas. Uma melhoria importante seria treiná-lo com uma base maior de dados e melhor distribuída nas faixas salariais, dessa forma seria possível estender a confiabilidade do modelo em salários mais altos.

Outra melhoria seria um trabalho mais elaborado sobre o pré-processamento dos dados, no qual poderia se tentar preencher valores vazios com informações extraídas das descrições, remover mais informações não relevantes dos títulos e descrições, entre outras ações.

Uma técnica mais avançada de representação de informação textual em vetores numéricos também poderia ser explorada, como o *Doc2Vec*. O aumento da qualidade dessa representação certamente poderia melhorar os resultados obtidos. Mesmo com a técnica *Word2Vec*, ao invés de utilizar o modelo pré-treinado do Google, uma possível melhoria seria treinar um modelo próprio para analisar descrições de vagas de emprego, que teria um vocabulário mais especializado nesse domínio.

Por fim, o modelo escolhido poderia ser melhor ajustado, com mais parâmetros sendo variados e com uma gama maior de valores. Esse ajuste mais fino não foi realizado por motivos de tempo e capacidade de processamento.

Referências

[1] Monster Jobs – Job Search, Career Advice & Hiring Resources.
https://www.monster.com

[2] Kaggle. US jobs on Monster.com.
https://www.kaggle.com/PromptCloudHQ/us-jobs-on-monstercom

[3] Tf-idf. Wikimedia Foundation.
https://en.wikipedia.org/wiki/Tf–idf

[4] Word2Vec. Wikimedia Foundation.
https://en.wikipedia.org/wiki/Word2vec

[5] word2Vec. Google Code Archive.
https://code.google.com/archive/p/word2vec/

[6] Kaggle. Job Salary Prediction.
https://www.kaggle.com/c/job-salary-prediction

[7] Jackman, S. & Reid, G. (2013). Predicting Job Salaries from Text Descriptions.
https://open.library.ubc.ca/cIRcle/collections/graduateresearch/42591/items/1.0075767