Introdução

Análise Geral

Ao realizar um describe(), que nos mostra as estatísticas descritivas gerais do dataset, podemos observar as duas colunas numéricas ID e SALARYNORMALIZED e suas características:

```
In [6]: job_treino.describe()
Out[6]:
                ID  SALARYNORMALIZED
count  2.447680e+05     244768.000000
mean   6.970142e+07      34122.577576
std    3.129813e+06      17640.543124
min    1.261263e+07       5000.000000
25%    6.869550e+07      21500.000000
50%    6.993700e+07      30000.000000
75%    7.162606e+07      42500.000000
max    7.270524e+07     200000.000000
```

Vemos que a média salarial dos cargos no geral é em torno de 34.122 euros, e desvio padrão de 17.640, ou seja, a diferença a partir da média para valores acima e abaixo desse resultado que nos mostra valores padrões/normais encontrados.

De 244.768 valores distintos, o valor mínimo do dataset é 5.000 euros, enquanto o valor máximo é de 40 vezes mais que o mínimo, 200.000 euros.

Análise de Colunas

ID:

O ID não será utilizado nas análises e como variável do modelo, tanto por ser um identificador único, quanto por terem inúmeras variáveis distintas.

Ele será utilizado mais à frente quando formos verificar o resultado do modelo com os arquivos compostos do dataset.

Uma observação feita é que o valor mínimo e máximo são respectivamente 12612628 e 72705235, o que mostra que pode ter sido uma amostra retirada de um conjunto maior de dados.

TITLE:

O título da vaga é uma coluna que podemos desconsiderar para a análise, pois ao ver que a coluna CATEGORY já possui as informações de qual área a vaga pertence e podemos utilizá-la para definir os valores de salário já.

FULLDESCRIPTION:

Ao ver a quantidade de valores únicos nessa coluna, percebemos que há 244768 valores únicos, o que torna a compreensão do modelo mais difícil, e além disso também podemos ver que não há uma relação entre a descrição do cargo com o salário, já que podemos ter tanto salários bons com descrições ruins e o inverso também.

É uma ótima opção realizar a remoção dessa coluna.

LOCATIONRAW:

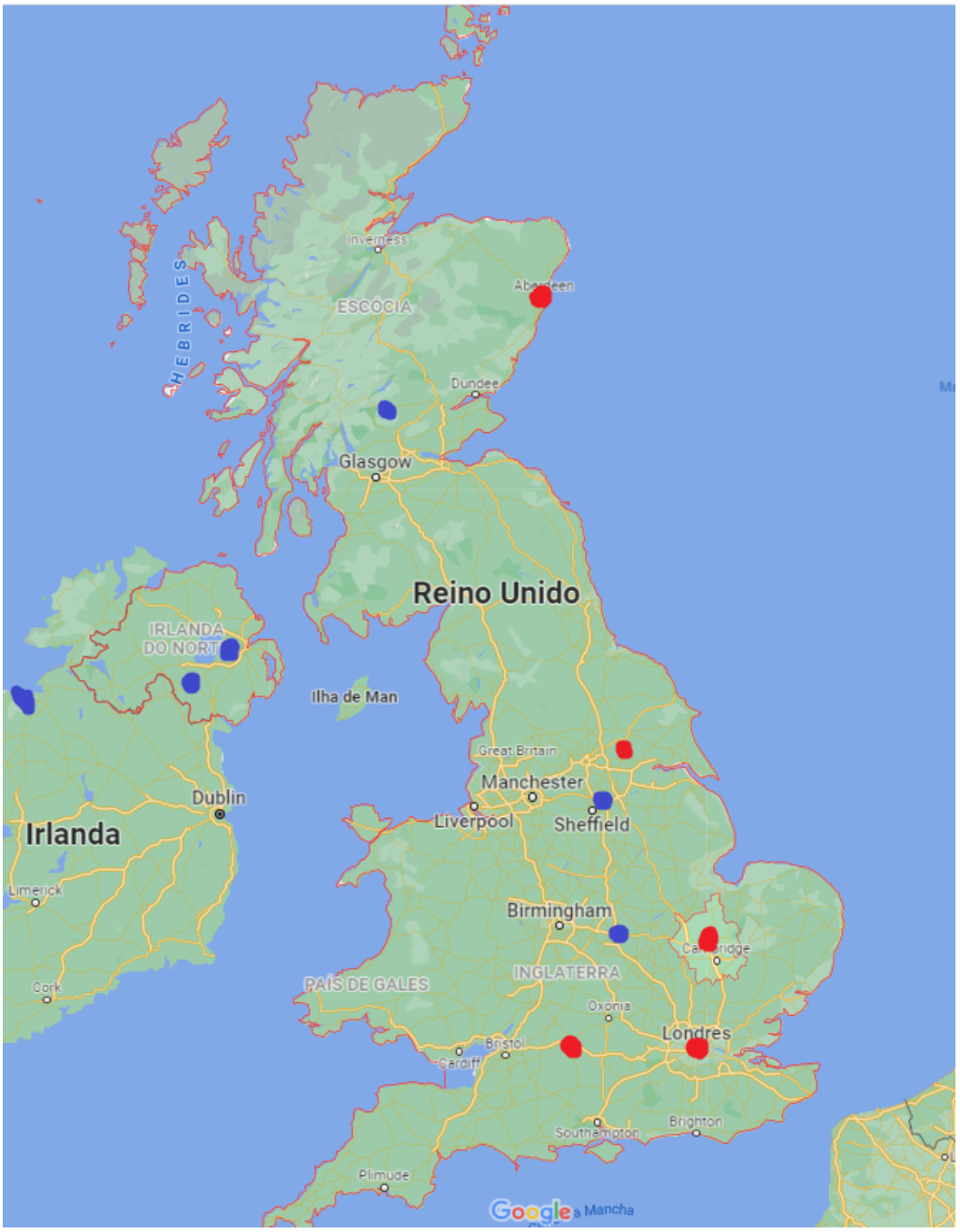
Assim como acontece com a coluna COMPANY, essa coluna tem o mesmo problema de alta cardinalidade entre os dados.

```
>>> len(job_treino["LOCATIONRAW"].value_counts())
19549
```

As cidades mostradas em suma maioria são do Reino Unido, porém podemos perceber também um problema de confiabilidade da coluna, já que encontramos dados repetidos em alta quantidade, como por exemplo CITY LONDON SOUTH EAST e LONDON SOUTH EAST, ambas têm a mesma localidade.

Podemos estar plotando as top 10 cidades de acordo com os índices de maior salário, e como vemos abaixo não há uma tendência de qual cidade se paga melhor:

Em azul podemos conferir cada cidade que apresenta os menores salários pagos para os funcionários, enquanto em vermelho mostra os maiores salários pagos.



Irei remover essa coluna pelos motivos acima e também por ter baixa representatividade de cada variável presente no dataset, tendo um valor mínimo e máximo respectivamente de 1 e 15986 localidades.

Como também há a coluna LOCATIONNORMALIZED então podemos acabar excluindo essa anterior, pois os valores nessa nova são mais limpos e mais agrupados, contendo apenas 2732 valores diversos.

LOCATIONNORMALIZED:

Como citado acima, utilizaremos a coluna para poder atribuir ao modelo, pois os valores estão já padronizados e agrupados pela cidade mais comum de cada linha.

CONTRACTTYPE:

Há dois tipos de valores presentes na coluna ao se realizar um value_counts, “FULL_TIME” e “PART_TIME”, ambos com respectivos valores 57538 (23%) e 7904 (3%).

Porém ao realizar uma verificação de valores NaN é observado que mais da metade dos dados (73%) apresentam valores NaN, o que torna problemático essa falta de valores.

A melhor opção que tive foi de substituir esses valores nulos pela string “NAO_IDENTIFICADO”, pois caso fossem removidas essas linhas, mais da metade do dataset estaria comprometido, e ao atribuir valores tanto de FULL_TIME ou PART_TIME eu poderia estar tendenciando uma variável mais que a outra.

```
>>> job_treino.isna().sum()
ID                       0
TITLE                    1
FULLDESCRIPTION          0
LOCATIONRAW              0
LOCATIONNORMALIZED       0
CONTRACTTYPE        179326
CONTRACTTIME         63905
COMPANY              32430
CATEGORY                 0
SALARYRAW                0
SALARYNORMALIZED         0
SOURCENAME               1
dtype: int64
```

CONTRACTTIME:

Nesta coluna temos dois tipos de variáveis, 151521 de valores sendo PERMANENT e 29342 sendo CONTRACT, o que significa respectivamente que são contratos do tipo permanente e apenas contratos com um tempo determinado, sendo representados por 81% e 19% da base.

Temos também uma quantidade de valores nulos um pouco alta em relação a quantidade de contratos, sendo 63905 valores nulos, o que é duas vezes mais que o tipo de contrato sendo CONTRACT.

Algumas resoluções que podemos ter seriam fazer o balanceamento entre as duas variáveis, porém a perda de dados do tipo PERMANENT é grande. Pode ser feito também a substituição dos valores nulos para CONTRACT e diminuir a diferença entre ambas as variáveis.

```
In [5]: job_treino["CONTRACTTIME"] = job_treino["CONTRACTTIME"].fillna("CONTRACT")
In [6]: job_treino["CONTRACTTIME"].value_counts()
Out[6]:
PERMANENT    151521
CONTRACT      93247
```

Temos agora um valor mais perto de 50/50 entre as duas variáveis, com uma porcentagem respectivamente de 61% e 39%.

COMPANY:

Nessa coluna observamos primeiramente quantos valores distintos temos de empresas, e temos como resultado 19856 variáveis diferentes.

```
>>> len(job_treino["COMPANY"].unique())
19856
```

Ao verificar a quantidade de valores únicos para cada empresa, vemos quantidades extremamente baixas como 2% da base no máximo de cada uma.

```
>>> job_treino["COMPANY"].value_counts()
UKSTAFFSEARCH                 4997
CVBROWSER                     2977
LONDON4JOBS                   2345
HAYS                          1869
JAM RECRUITMENT LTD           1241
```

Essa alta cardinalidade de valores nos induz a remover essa coluna, já que com altas quantias de variáveis o modelo possa não se sair tão bem, porém irei utilizar para metrificar se a companhia influencia negativamente mesmo ou não.

CATEGORY:

A categoria nos mostra os ramos de atividade, por exemplo área de TI, advocacia, médica etc…

Temos um valor máximo e mínimo de 38483 e 145, representando a diferença em porcentagem temos 99,7% a 0,3%.

Podemos estar agrupando os últimos valores e inserir a variável OUTROS, porém percebemos que já existe essa variável denominada como OTHER/GENERAL JOBS representando 44% da base total.

Manteremos a CATEGORY sem alterações até a realização do modelo para evitar muitas variações.

SALARYRAW:

Do mesmo modo que ocorre em LOCATIONRAW e LOCATIONNORMALIZED, aqui em salários temos o range do que a pessoa com contrato permanente e contrato recebem tanto anual quanto mensalmente.

Podemos optar por permanecer apenas com a coluna SALARYNORMALIZED, que além de ser o target de nosso estudo, também prioriza a média salarial de ambas as categorias de contrato, o que também mantém a coluna como um tipo único como INT invés de precisarmos convertê-la já que ela vem como STRING.

SALARYNORMALIZED:

Optamos por permanecer com essa coluna já que ela é a variável target do modelo de estudos normalizada.

SOURCENAME:

Temos 167 sites diferentes de classificados, com valores mínimos de 1 e máximo de 48149 vagas para cada classificado.

Como há essa grande diferença entre a quantidade de vagas para cada site, faremos um range de apenas 20 das empresas com mais vagas disponíveis.

```
In [22]: job_treino["SOURCENAME"].value_counts()[:20].sum()
Out[22]: 180600
```

Ao realizar essa diminuição, vemos que a quantidade de linhas não diminui drasticamente, o que é um bom sinal, manteremos dessa forma reduzida, passando de 167 classificados para apenas 20.

Tratativas Gerais

Após o tratamento individual de cada coluna, é realizado o tratamento para poder inicializar a fase de modelagem dos dados.

Realizamos a normalização dos dados e alterando o valor deles de categórico para número, colocando um range entre 0 e 1 para os dados não serem tão discrepantes um dos outros.

Através do SimpleImputer, inserimos em todos os valores faltantes valores de variáveis mais frequentes de cada coluna, preenchendo assim qualquer resquício de Nan que tenhamos deixado.

```
Out[33]:
        LOCATIONNORMALIZED  CONTRACTTYPE  CONTRACTTIME  CATEGORY  SOURCENAME
0                    660.0           1.0           1.0       8.0         2.0
1                    870.0           1.0           1.0       8.0         2.0
2                    943.0           1.0           1.0       8.0         2.0
3                   2040.0           1.0           1.0       8.0         2.0
4                   2040.0           1.0           1.0       8.0         2.0
                       ...           ...           ...       ...         ...
244763              2056.0           1.0           0.0      25.0         7.0
244764              2056.0           1.0           0.0      25.0         7.0
244765              2056.0           1.0           0.0      25.0         7.0
244766              2328.0           1.0           0.0      25.0         7.0
244767              1037.0           1.0           1.0      25.0         7.0

[180600 rows x 5 columns]
```

É realizada a separação dos dados de treino e teste para fazermos a validação e metrificação dos dados.

E então realizamos um tuning dos hiperparâmetros para que tenhamos o melhor valor de parâmetro que possa ser passado para o modelo escolhido que foi o RandomForestRegressor.

Utilizei o RandomForestRegressor pelo simples fato de ser um algoritmo mais básico e mais completo ao se tratar de previsão de modelos de regressão.

Foi construído um grid e com a ajuda do RandomizedSearchCV podemos estimar com uma maior certeza qual valor de parâmetro é o melhor para o modelo.

```
np.random.seed(1)
grid = {
    "n_estimators": np.arange(10, 100, 10),
    "max_depth": [None, 3, 5, 10],
    "min_samples_split": np.arange(2, 20, 2),
    "min_samples_leaf": np.arange(1, 20, 2),
    "max_features": [0.5, 1, "sqrt", "auto"],
    "max_samples": [10000, 12000, 15000, 20000]
}

rs_model = RandomizedSearchCV(RandomForestRegressor(random_state=1), param_distributions=grid)
```

O resultado podemos conferir logo abaixo:

{'n_estimators': 50, 'min_samples_split': 6, 'min_samples_leaf': 1, 'max_samples': 20000, 'max_features': 'auto', 'max_depth': None}

Com esses valores podemos substituir em nosso modelo original e então ver o resultado da predição corretamente.

Métricas e Resultados

Para calcular quão bem o modelo foi, iremos testar algumas métricas de regressão, entre elas a MSE (Mean Squared Error), R² (R-Squared), MAE (Mean Absolute Error) e MAPE (Mean Absolute Percentage Error).

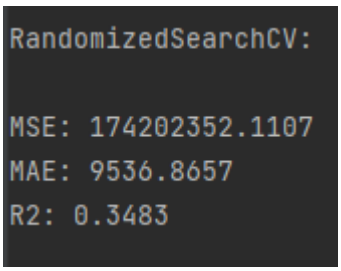
1. MSE: É a diferença entre o valor predito pelo modelo e o valor real elevado ao quadrado, quanto maior este valor pior.

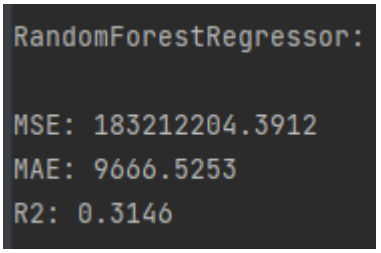
2. R²: Nos diz a porcentagem da variância que pôde ser prevista, ou seja, nos diz o quão "próximo" as medidas reais estão do nosso modelo.

3. MAE: A média das distâncias entre valores preditos e reais.

4. MAPE: Medida para definir a acurácia do modelo.

Com as colunas LOCATIONNORMALIZED, CONTRACTTYPE, CONTRACTTIME, CATEGORY e SOURCENAME, conseguimos obter as seguintes métricas tanto usando o modelo puro quanto usando os parâmetros obtidos do RandomizedSearchCV:



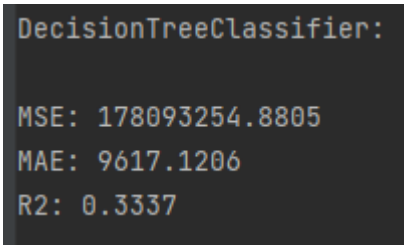


Para ambos os resultados de modelos as métricas foram péssimas, com valores muito baixos de R2 (tanto 34% quanto 31%), o MSE está altíssimo para ambos sendo que o ideal é quanto mais próximo do zero melhor.

Agora iremos ajustar algumas variáveis e alterar o modelo para verificarmos se o resultado pode melhorar ou se continuará ruim.

MODELOS:

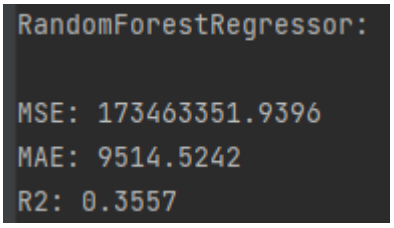
Foi testado além do RandomForestRegressor também o DecisionTreeRegressor, porém o resultado dele aplicando o RandomizedSearchCV não foi tão satisfatório quanto o resultado



do modelo anterior citado, como podemos ver ao lado, os resultados foram próximos porém abaixo do que esperávamos.

Manteremos o Random Forest como modelo principal do projeto então.

TREINO E TESTE:

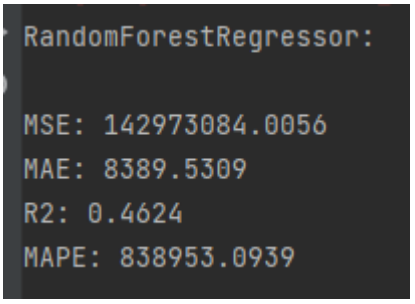


Para os parâmetros do train_test_split inserimos alguns valores para o tamanho do teste, e o que melhor deu resultado foi a separação de 85% da base para treino enquanto 15% fica sendo para teste.

Como o R² acabou tendo uma performance melhor inclusive outras métricas, acabamos optando por manter dessa forma a separação.

COLUNAS:

Irei inserir algumas colunas para ver a influência no modelo, primeiramente começando pela coluna COMPANY:



Percebemos o aumento de 5% do R² e uma diminuição de outras métricas também.

Vemos um aumento no resultado inserindo a coluna TITLE no dataset também, manteremos ambas as colunas pelo aumento do valor.

Conclusão

Com o resultado de um R² de 46% e um MSE e MAE de valores altíssimos podemos ver que a performance do modelo não é tão boa, porém podemos fazer uma média de valores máximos e mínimos para um salário e assim acertar sua média de preço.

Temos também os valores mínimos e máximos para assim formar o range de salário da vaga, já que conseguimos o valor do erro padrão da análise.

Podemos então estar informando ao cliente o range dos valores junto com a performance do modelo.