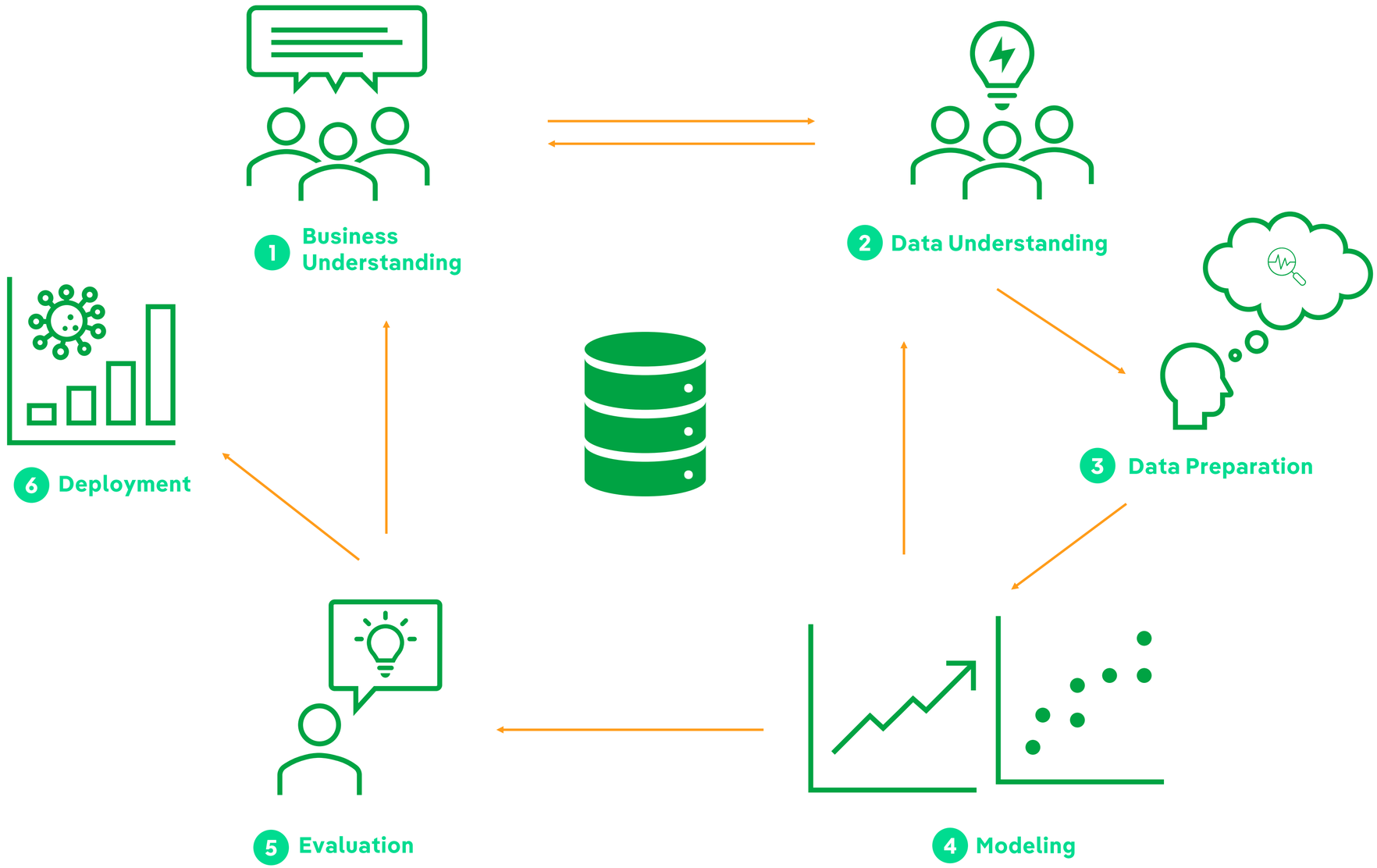
APRESENTAÇÃO

CASE

TAXI TRAJECTORY PREDICTION

13/12/2023

Metodologia



Premissas

Desafio: Existem serviços que despacham pedidos de táxi e seria interessante prever se o local de desembarque será próximo no próximo embarque, facilitando a operação com um todo. Devido a falta de compartilhamento de alguns trajetos e mudanças de sistemas, hoje é difícil para a operadora conseguir contornar essas dificuldades. Logo, foi proposto o desenvolvimento de uma aplicação capaz de predizer o local de destino mais fidedigno.

Os dados disponibilizados contém:

- **Mais de 1,7M de registros, com 9 colunas sendo:**
- **TRIP_ID**: ID da viagem (único)
- **CALL_TYPE**: identifica o tipo da corrida, sendo:
  - **A**: se foi despachada pela central;
  - **B**: se surgiu a partir de um ponto de táxi;
  - **C**: aleatória de alguma rua;
- **ORIGIN_CALL**: identificador único caso haja informação do cliente, solicitado pela central (CALL_TYPE = A). Se for diferente disso é Nulo;
- **ORIGIN_STAND**: identificador único do ponto de táxi (Nulo se CALL_TYPE != B)
- **TAXI_ID**: ID do taxista
- **TIMESTAMP**: tempo em segundos (UTC)
- **DAYTYPE**: B se for feriado, C se for pós feriado e A se não for nenhum dos 2 anteriores;
- **MISSING_DATA**: FALSO se todas as coordenadas estiverem completas, caso contrário, VERDADEIRO;
- **POLYLINE**: coordenadas do trajeto

Planejamento da solução

Saída

- Métricas regressão: R^2 e RMSE
- CSV com as predições
- Comparativo de modelos

Processamento

- Entender o dataset
- Testar alguns dados para verificar faltantes
- Mapa dos pontos geográficos para trazer algum insight
- Criar variáveis derivadas de uma ou mais colunas
- Verificar dados categóricos quanto a concentração em uma classe
- Filtrar outliers caso exista
- EDA + Engenharia de features + preparação

Entrada

- Dados da plataforma
- Treino: > 1M de dados
- Teste: usá-lo no final
- Dados em CSV (+ 1GB)

Entendimento

Da carga dos dados observou que:

- ORIGIN_CALL com 78% NaN;
- ORIGIN_STAND com 52% NaN;
- Colunas quase que com todos os tipos corretos, exceto por POLYLINE (object);
- Cada registro de coordenadas POLYLINE possui tamanhos diferentes;
- Coordenadas iniciais e finais vazias em alguns casos;

Realizou-se o tratamento, sendo:

- Realizar uma partição nos 2 primeiros dígitos e 2 últimos de POLYLINE e removê-los caso = “[]”;
- Transformar TIMESTAMP na coluna DATE via datetime com UTC = TRUE;
- A partir de DATE, criar uma nova “DATE2” com apenas YYY-MM-DD;
- As colunas com TRUE/FALSE passar a ser 1 ou 0, devido aos modelos utilizarem dados numéricos;
- As colunas com NaN values foram preenchidas com 0, e os demais valores únicos com 1. Isso porque não queremos saber o cliente, mas sim se TEVE INFORMAÇÃO DE CLIENTE ou NÃO;
- Colunas YEAR e MONTH derivadas de ‘DATE’ (facilitar plots)
- Devido a tamanhos diferentes dentro de POLYLINE, obteve-se somente a LAT/LONG inicial e final

Exemplos:

```
# verify if we have None init lat/long
df['polyline'].str[:2].unique()
```

```
array(['[[', '[]'], dtype=object)
```

```
print(df.shape)
df = df.loc[df['polyline'].str[:2] != '[]']
df = df.loc[df['polyline'].str[-2:] != '[]']
print(df.shape)
```

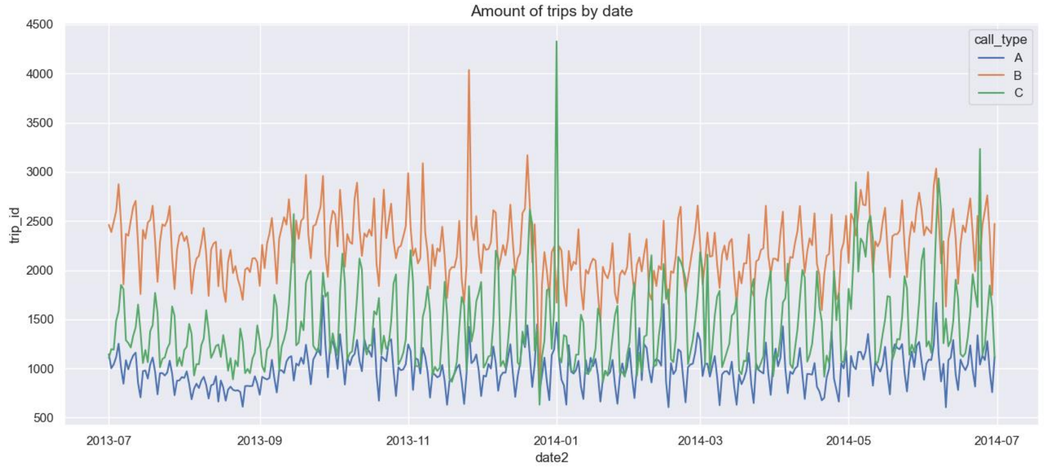
```
(1710670, 9)
(1704769, 9)
```

```
# change space of origin_stand
df['int_origin_stand'] = 1
df.loc[df['origin_stand'].isna(),'int_origin_stand'] = 0

# change space of origin_call
df['int_origin_call'] = 1
df.loc[df['origin_call'].isna(),'int_origin_call'] = 0


# change space of missing_data
df['int_missing_data'] = 1
df.loc[(df['missing_data'] == False) | (df['missing_data'] == 'False'),
       'int_missing_data'] = 0
```

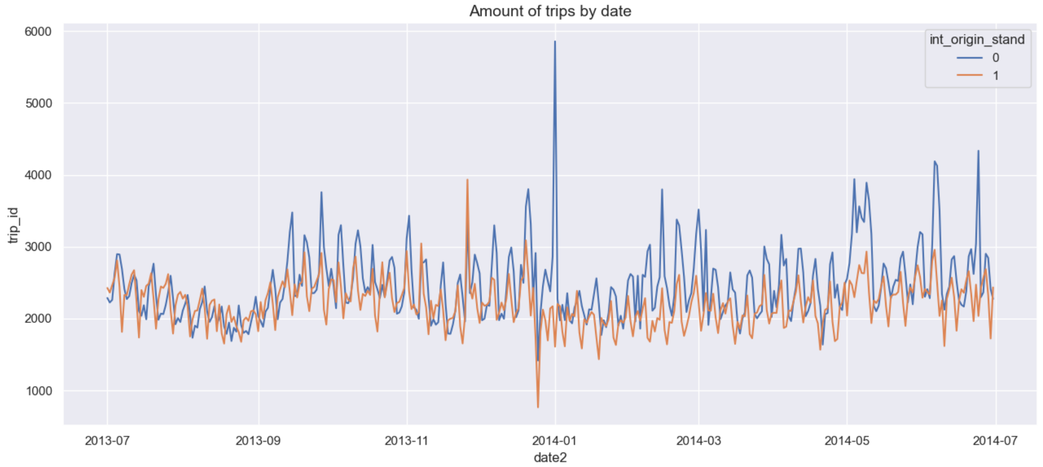
```
# change to UTC date format
df['date'] = pd.to_datetime(df['timestamp'], unit='s', utc=True)

# just keep YYYY-MM-DD
df['date2'] = pd.to_datetime(df['date'].astype(str).str[:10],format = '%Y-%m-%d')
```

```
vetor_coord = df['polyline'].values
lat_init = []
lat_end = []
long_init = []
long_end = []

for i in vetor_coord:
    string_vetor = re.sub(r'[\[\]\|]', '', i)
    lat_init.append(string_vetor.split(',')[1])
    lat_end.append(string_vetor.split(',')[-1])
    long_init.append(string_vetor.split(',')[0])
    long_end.append(string_vetor.split(',')[-2])


df['lat_init'] = lat_init
df['long_init'] = long_init
df['lat_end'] = lat_end
df['long_end'] = long_end
for i in ['lat_init','lat_end','long_init','long_end']:
    df[i] = df[i].astype(float)
```

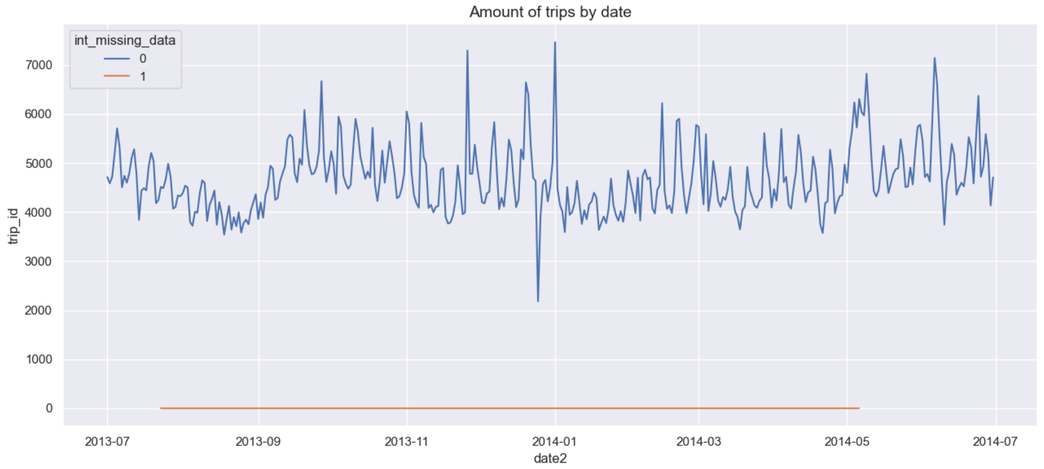
EDA



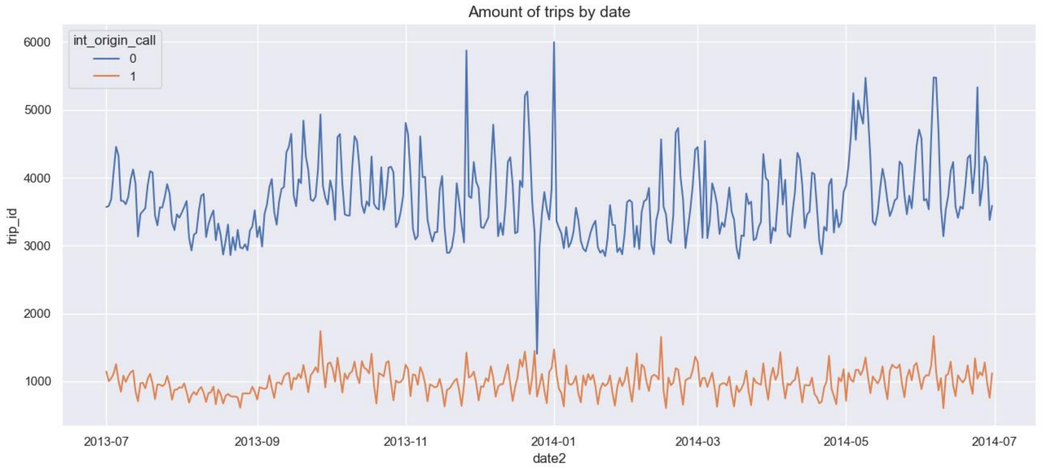
CALL_TYPE B é o que prevalece



Trajeto sem um ponto de táxi tem mais corridas

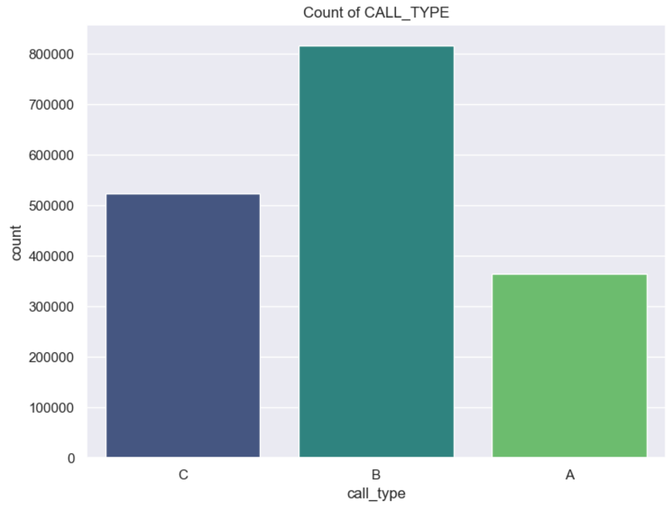


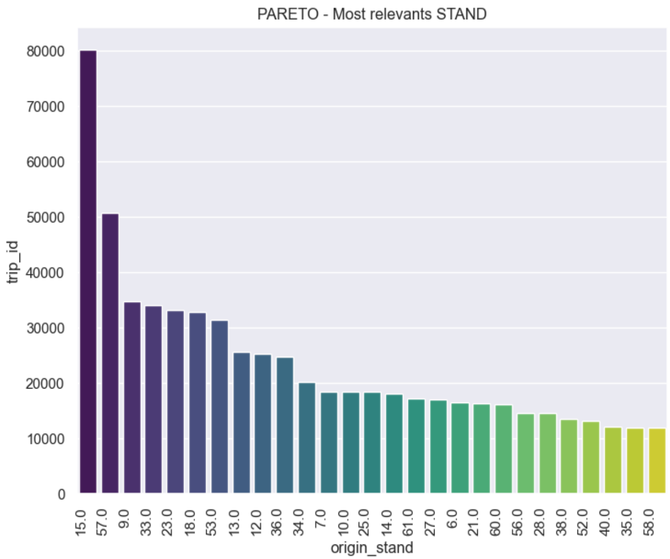
Poucos dados possuem coordenadas faltantes



A maioria das corridas não possui informação do cliente

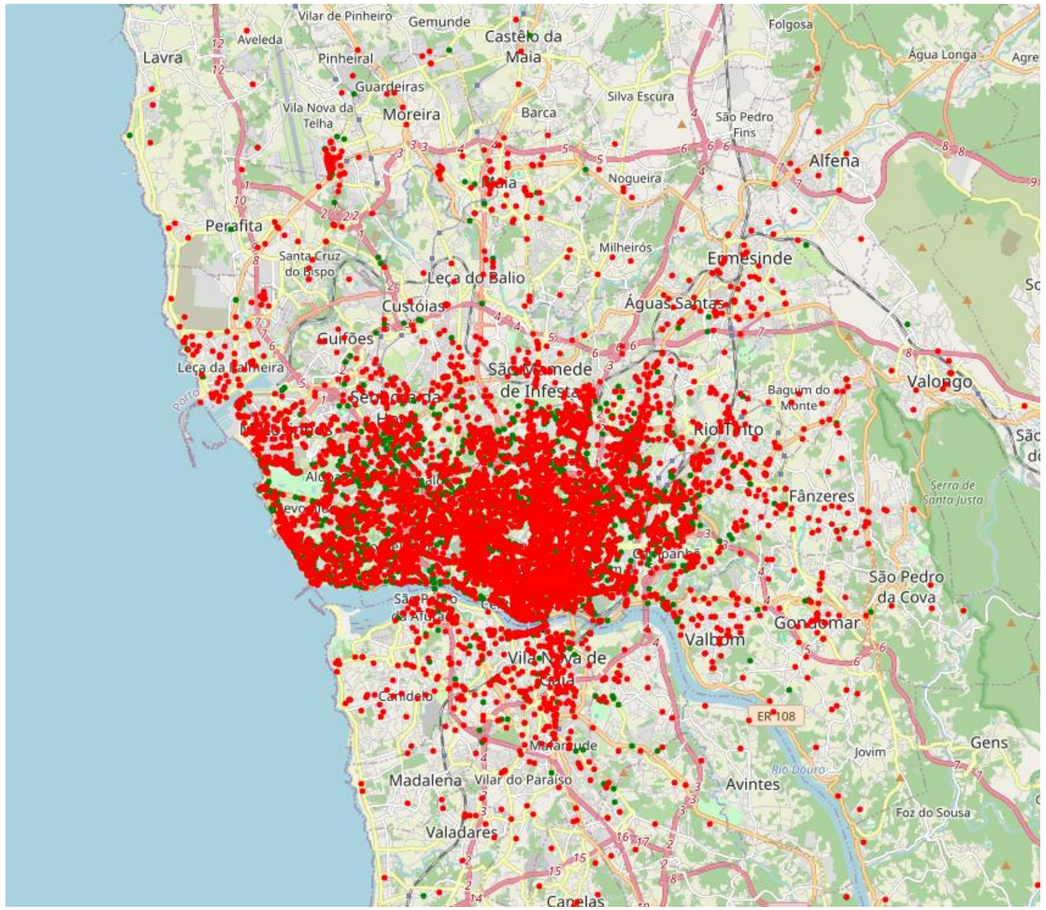
EDA





Corridas

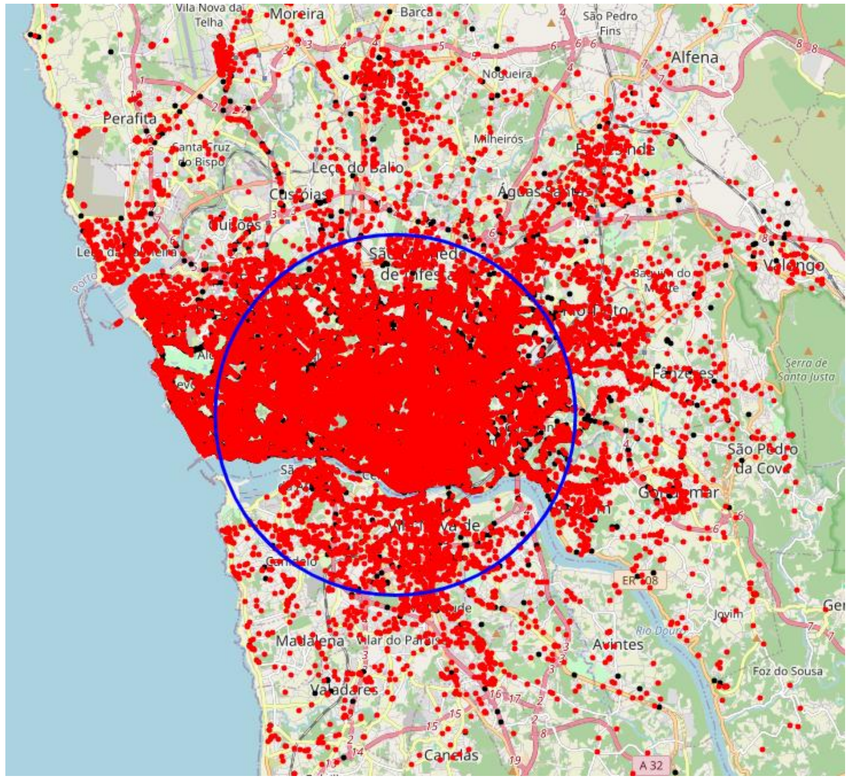
- **80%** concentradas em **27 dos 63** pontos de táxi
- Destaque para **15 e 57** com mais de **50k** de corridas
- **CALL_TYPE** = **B** é o que prevalece

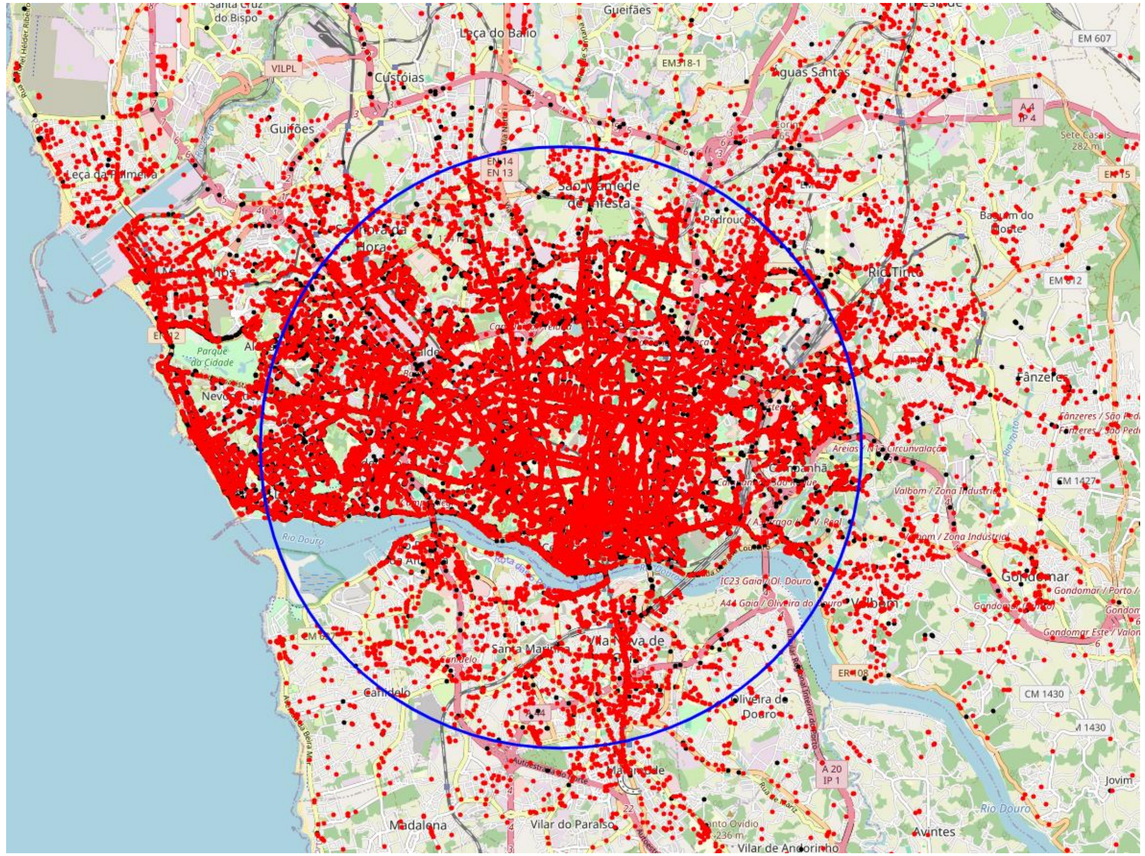


Amostra de 10k de corridas

- Concentrando em trajetos próximos a região central
- **Pontos verdes**: coordenadas de início
- **Pontos vermelhos**: coordenadas fim

EDA



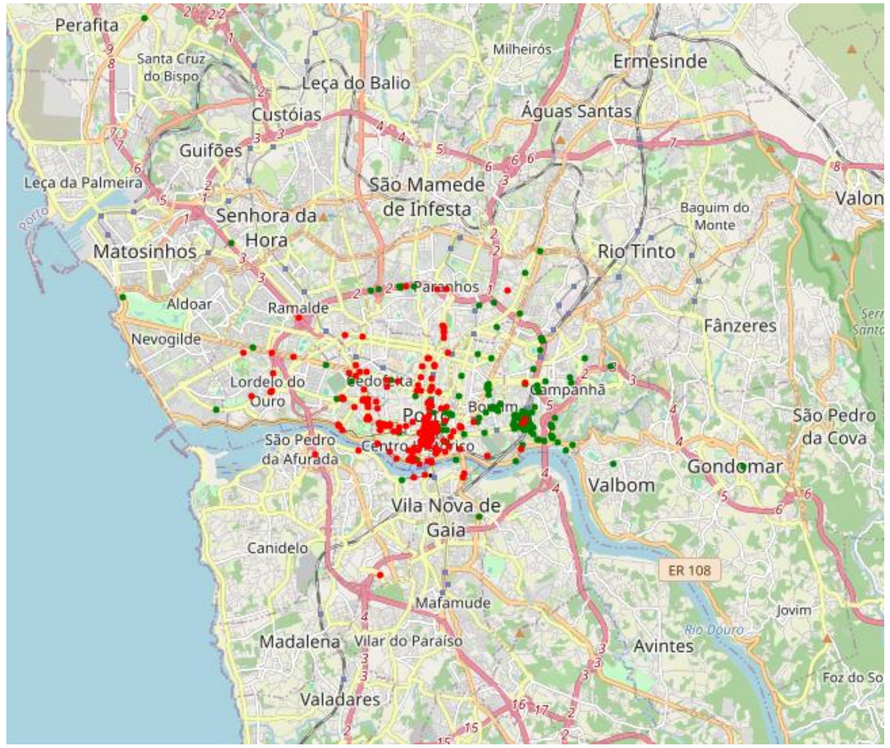


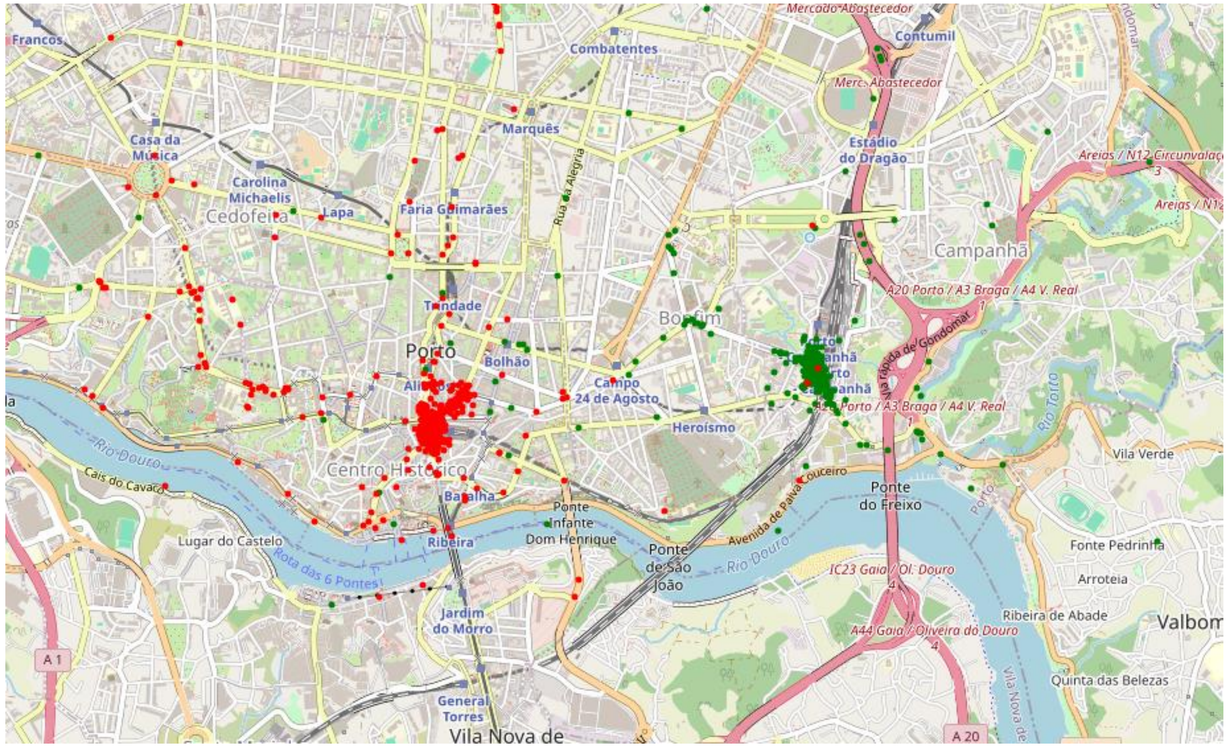
60k de dados*

- 80% das corridas são finalizadas em até 4,5km do centro
- 50% das corridas são finalizadas em até 2,5km do centro
- **Pontos pretos**: coordenadas de início
- **Pontos vermelhos**: coordenadas fim

***Validado com a base completa também**

EDA

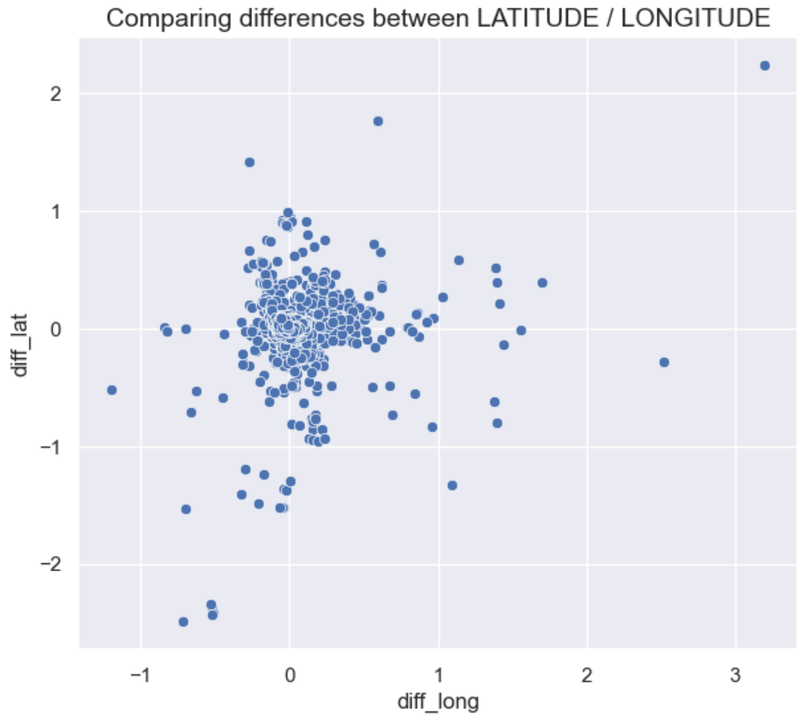


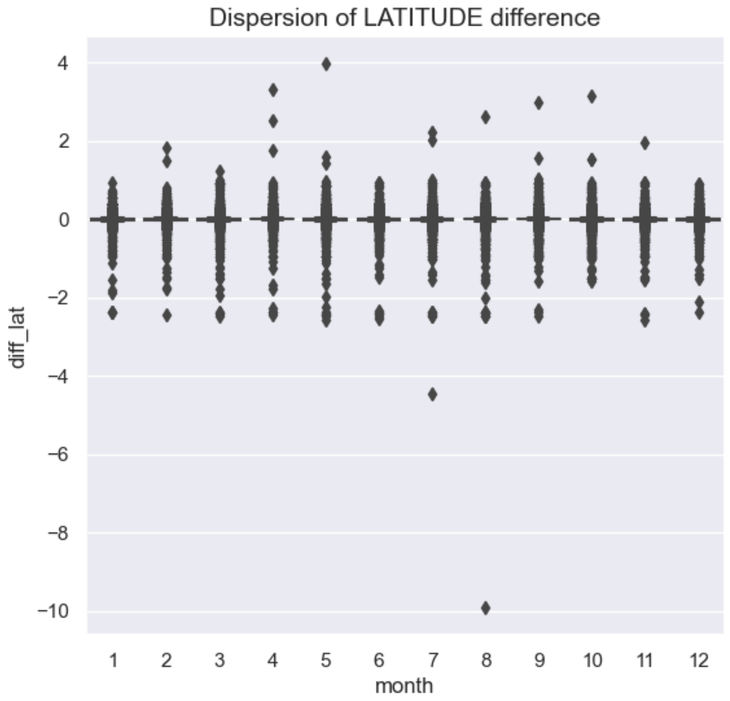


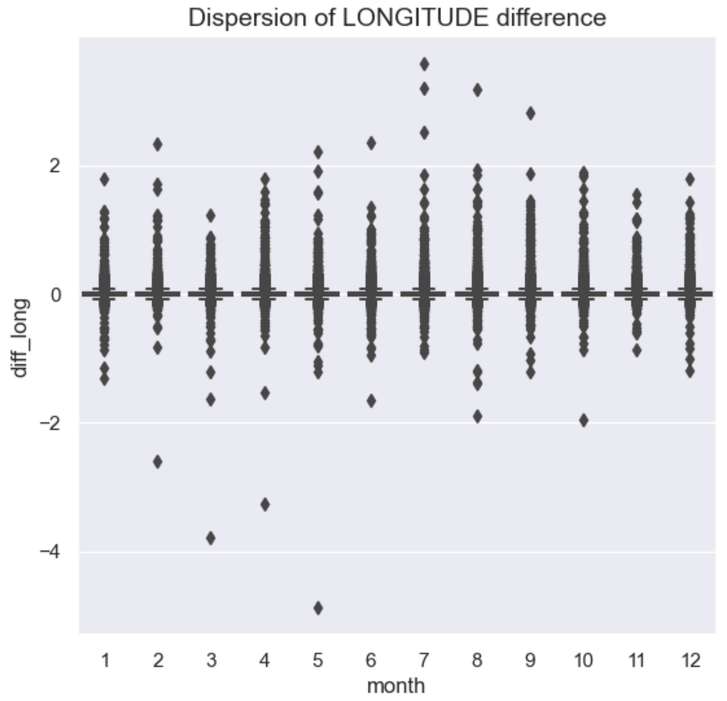
Coordenadas iniciais PONTO 15 e 57

- Região central de Porto
- Estão conectados com ferrovias e rodovias principais
- Entende-se que a maior parte das corridas iniciam desses pontos e permanecem dentro da região

Entendimento + Preparação







Amostra de 200k de dados

- Há alguns dados dispersos das diferenças de lat e long
- Mostra que a concentração das corridas está na parte central de Porto
- Poucos são os casos afastados da área central

1,7M de dados

- Apesar da dificuldade em visualizar os limites, mesmo segregando por mês, é possível notar que há outliers, sendo os ponto mais afastados da região central

Modelagem

Foram feitos os seguintes procedimentos:

- Criar as variáveis de diferença entre as latitudes e longitudes iniciais e finais;
- Remover os outliers via interquartil (IQR) das diferenças acima;
- Criar variáveis “dummies” para CALL_TYPE;
- Selecionar as colunas numéricas para o modelo, sendo:
- 'int_origin_stand', 'int_origin_call', 'int_missing_data’, 'ct_A', 'ct_B’, 'ct_C’, 'lat_init’, 'long_init’, 'diff_lat’, 'diff_long’, 'lat_end', 'long_end’;
- Amostrar o dataset caso estouro de memória;
- Segregar entre Treino e Teste com tamanhos 75% e 25%, respectivamente;
- Variável resposta: [latitude_final, longitude_final]
- Utilizar o MultiOutputRegressor( ) do Scikit-learn para lidar com a saída (mais de 1 dimensão)
- Testar os modelos para comparativo entre métricas: Random Forest, XGB, Dummy, Linear, Lasso, Ridge e RANSAC

Exemplos:

```
iqr_lat = df['diff_lat'].quantile(0.75) - df['diff_lat'].quantile(0.25)
min_lat = df['diff_lat'].quantile(0.25)  - (1.5*iqr_lat)
max_lat = df['diff_lat'].quantile(0.75)  + (1.5*iqr_lat)

iqr_long = df['diff_long'].quantile(0.75) - df['diff_long'].quantile(0.25)
min_long = df['diff_long'].quantile(0.25)  - (1.5*iqr_long)
max_long = df['diff_long'].quantile(0.75)  + (1.5*iqr_long)
```

```
# dummies for call_type
df = pd.get_dummies(df,columns=['call_type'],prefix='ct', drop_first=False)
```

```
X = df[cols_modelo].sample(frac = 0.5)

y = X[['lat_end', 'long_end']]

X = X.drop(['lat_end', 'long_end'], axis = 1)
```

Model	RMSE	R2
LinearReg	3.034204e-14	1.000000
RANSAC	1.733783e-11	1.000000
Ridge	1.319802e-03	0.997979
Dummy	4.455276e-02	-0.000008
Lasso	4.455276e-02	-0.000008
XGB	7.376537e-02	-0.718443

Modelagem

Linear
Haversine(lat_linear, long_linear, lat_inicio, long_inicio)

Real
Haversine(lat_fim, long_fim, lat_inicio, long_inicio)

~0,00001 KM
Soma das diferenças

Dummy
Haversine(lat_dummy, long_dummy, lat_inicio, long_inicio)

~0,17 KM
Soma das diferenças

Modelo escolhido
Regressão Linear

Motivo
Melhor performance
simplicidade

Métricas
RMSE: 3,03 x 10^(-14)

Próximos passos

- **Validar a possibilidade de usar o estudo para identificar motoristas que irão finalizar corridas próximas a uma que estiver pra ser iniciada (indicar o mais próximo);**

- **Buscar o valor das corridas para analisar a possibilidade de melhorar a tarifa com base na probabilidade da corrida ser finalizada em zonas dentro dos raios mostrados anteriormente (2,5 km e 4,5 km);**

- **Estimular serviços baseados no despacho, visto que corridas a partir de pontos pré-definidos ou ruas aleatórias são os tipos mais concentrados.**