PROJETO 2 – MICROSOFT AZURE MACHINE LEARNING

Projeto da Formação Cientista de Dados da Data Science Academy

Grupo Bimbo Inventory Demand

In this competition, Grupo Bimbo invites Kagglers to develop a model to accurately forecast inventory demand based on historical sales data. Doing so will make sure consumers of its over 100 bakery products aren't staring at empty shelves, while also reducing the amount spent on refunds to store owners with surplus product unfit for sale.

Data fields

- Semana — Week number (From Thursday to Wednesday)
- Agencia_ID — Sales Depot ID
- Canal_ID — Sales Channel ID
- Ruta_SAK — Route ID (Several routes = Sales Depot)
- Cliente_ID — Client ID
- NombreCliente — Client name
- Producto_ID — Product ID
- NombreProducto — Product Name
- Venta_uni_hoy — Sales unit this week (integer)
- Venta_hoy — Sales this week (unit: pesos)
- Dev_uni_proxima — Returns unit next week (integer)
- Dev_proxima — Returns next week (unit: pesos)
- Demanda_uni_equil — Adjusted Demand (integer) (This is the target you will predict)

Equipamentos e software de apoio

Utilizamos para esse projeto um computador HP com CORE i5, 16 M de RAM, RStudio, Microsoft Azure Machine Learning, Lab Notebook com Python 3 e o SQLite para algumas análises dos arquivos que continham até 26 milhões de registros.

ANÁLISE EXPLORATÓRIA DOS DADOS

Verificamos que algumas variáveis fornecidas não estavam contidas no arquivo com os dados de teste fornecidos – test.csv

Sendo assim consideramos as seguintes variáveis:

- id Semana Agencia_ID Canal_ID Ruta_SAK Cliente_ID Producto_ID

A variável id por ser apenas um identificador dos registros será descartada na criação do modelo preditivo.

Através do módulo Filter Based Feature Selection do Microsoft Azure ML identificamos as seguintes correlações das variáveis preditoras com a variável target de Demanda:

Demanda	Canal_ID	Producto_ID	Ruta_SAK	Cliente_ID	Agencia_ID
1	0.211166	0.119778	0.106202	0.053182	0.015435

A variável preditora Canal_ID foi a que apresentou a mais forte correlação com a Demanda, mas utilizamos todas as 5 variáveis preditoras no nosso modelo.

Como foram fornecidas nove semanas no arquivo train.csv e apenas duas semanas no arquivo test.csv, fizemos o agrupamento dessa variável considerando o valor médio das 9 semanas para a estimativa da variável target Demanda utilizando a linguagem R:

```
install.packages("readr")

install.packages("dplyr")

install.packages("data.table")

library(readr)

library(dplyr)

library(data.table)

Train <- fread("train.csv")

new_Train <- Train %>%

 group_by(Agencia_ID, Canal_ID, Ruta_SAK, Cliente_ID, Producto_ID) %>%

 summarise(Demanda = mean(Demanda_uni_equil))
```

Salvamos o resultado:

```
write_csv(new_Train,"Train_demanda_media")
```

Utilizando a linguagem Python no Lab Notebook, escolhemos o Algoritmo de Regressão XGBRegressor e a métrica de avaliação solicitada no Kaggle Root Mean Squared Logarithmic Error.

```
# Import dos módulos
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_log_error
from xgboost import XGBRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

# Carregando os dados

dados = read_csv("Train_demanda_media")
array = dados.values

# Separando o array em componentes de input e output
X = array[:,0:5]
Y = array[:,5]

scaler = StandardScaler()
X = scaler.fit_transform(X)

# Divide os dados em treino e teste
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.3)

# Criando o modelo
modelo = XGBRegressor()

# Treinando o modelo
modelo.fit(X_train, Y_train)
```

```
# Fazendo previsões
Y_pred = modelo.predict(X_test)

# Como algumas previsões no Y_pred são negativas e o MSLE não permite valores negativos no seu cálculo,
# consideramos 0 para toda previsão negativa e arredondamos o valor de saída

Y_pred = np.array(Y_pred)

def zeraneg(i):
  if i < 0:
    return 0
  else:
    return i

Y_pred = list(map(zeraneg, Y_pred))

Y_pred = [round(value) for value in Y_pred]

# Resultado
msle = mean_squared_log_error(Y_test, Y_pred)
print("O MSLE do modelo é:", msle)
```

[12:27:14] WARNING: C:\Users\Administrator\workspace\xgboost-win64_release_1.0.0\src\gbm\gbtree.cc:138: Tree method is automatically selected to be 'approx' for faster speed. To use old behavior (exact greedy algorithm on single machine), set tree_method to 'exact'.

O **MSLE** do modelo é: **0.3982225796934346**

```
# Salvando o modelo
import pickle
arquivo = 'modelo_regressor_final.sav'
pickle.dump(modelo, open(arquivo, 'wb'))

# Fazendo as previsões
```

```
dados = read_csv("test.csv")

array = dados.values

X = array[:,2:7]

scaler = StandardScaler()

X = scaler.fit_transform(X)

Y_pred = modelo.predict(X)

Y_pred = np.array(Y_pred)


def zeraneg(i):
   if i < 0:
      return 0
   else:
      return i


Y_pred = list(map(zeraneg, Y_pred))


Y_pred = [int(round(value)) for value in Y_pred]


Previsoes = pd.DataFrame(Y_pred)

Previsoes.to_csv('Sample_subimssion_PSTT.csv')
```

CONCLUSÃO

O modelo final do algoritmo Algoritmo de Regressão XGBRegressor apresentou um ótimo aprendizado comprovado pela métrica MSLE de **0,3982** e que ainda pode ser melhorado com os ajustes dos Hyperparametros, mas que para isso aconselhamos um outro equipamento com maior capacidade computacional.