Abaixo estarei mostrando como foi realizado o estudo de previsão de preço de abacates do
dataset Avocado Prices que está presente no Kaggle pelo link abaixo e também explicarei o código que foi feito.

Kaggle Dataset: https://www.kaggle.com/neuromusic/avocado-prices

1- Importando os arquivos necessários

Utilizei as seguintes bibliotecas para realizar o estudo:

```
import warnings
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import pickle

from sklearn.preprocessing import OrdinalEncoder
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score,
max_error

warnings.simplefilter("ignore")
pd.options.display.max_columns = None
```

2- Importando o arquivo

Há apenas um arquivo csv na competição chamado “avocado.csv”. Realizo a importação dele e caso não consiga fazer o mesmo, irá entrar na exceção.

```
try:
    arquivo = pd.read_csv("avocado.csv", sep=",")

except FileNotFoundError:
    print("ARQUIVO NAO FOI ENCONTRADO")
```

```
except Exception as e:
    print(f"ERRO AO SUBIR ARQUIVO: {e}")
```

Após realizar todos os tratamentos e limpeza dos dados, realizei a separação de 30% da base para teste. Utilizei a função sample() para que pudesse pegar as linhas aleatoriamente.

```
arquivo_teste = arquivo.sample(frac=0.3, replace=True, random_state=1)
arquivo_teste.to_csv(r"C:\Users\Mayara Lopes\Desktop\GitHub\machine_learning_projects\Avocado\teste.csv", index=False)
```

3- Limpeza e Tratamento

Limpeza geral como substituir pontuação, remoção de acentos e pontos, e padronizar todo
o dataset para Uppercase. Padrão que utilizo normalmente em meus datasets.

```
arquivo = arquivo.replace(".", "").replace(",", ".")
arquivo.columns = arquivo.columns.str.upper()
```

Além de outros tratamentos como por exemplo a substituição da palavra “conventional” e “organic” para 1 e 0 respectivamente:

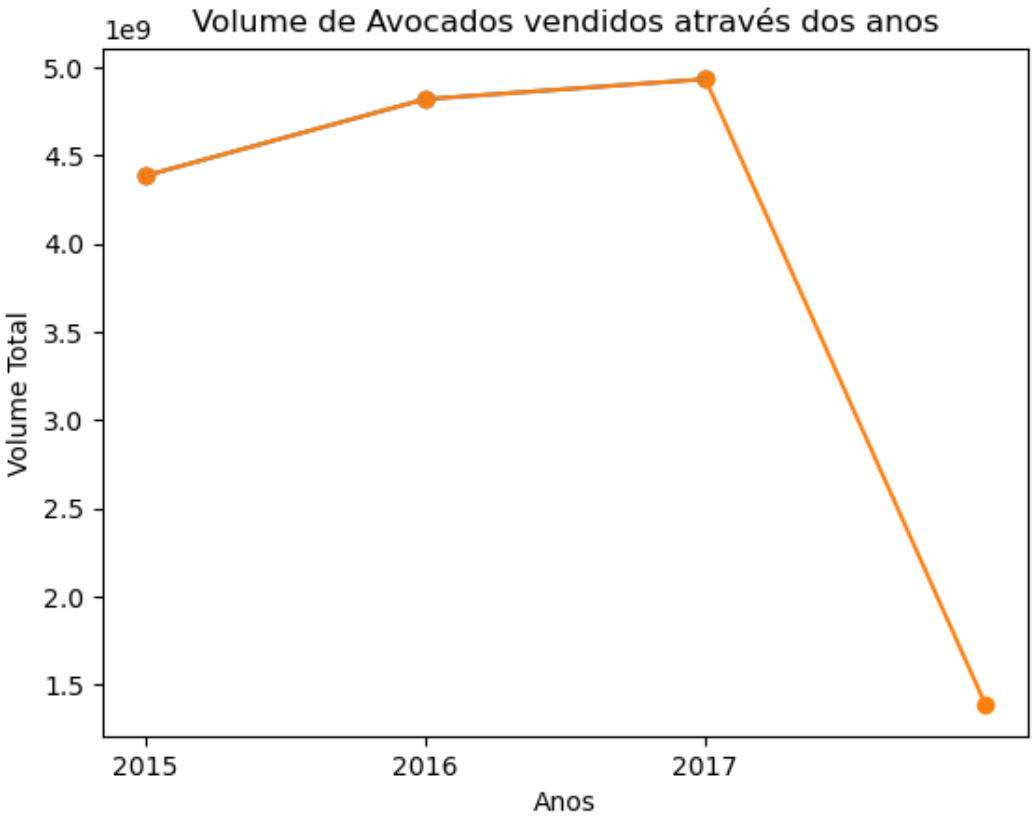
```
arquivo["TYPE"] = arquivo["TYPE"].replace("conventional", 1).replace("organic", 0)
```

4- Análises para tomadas de decisões

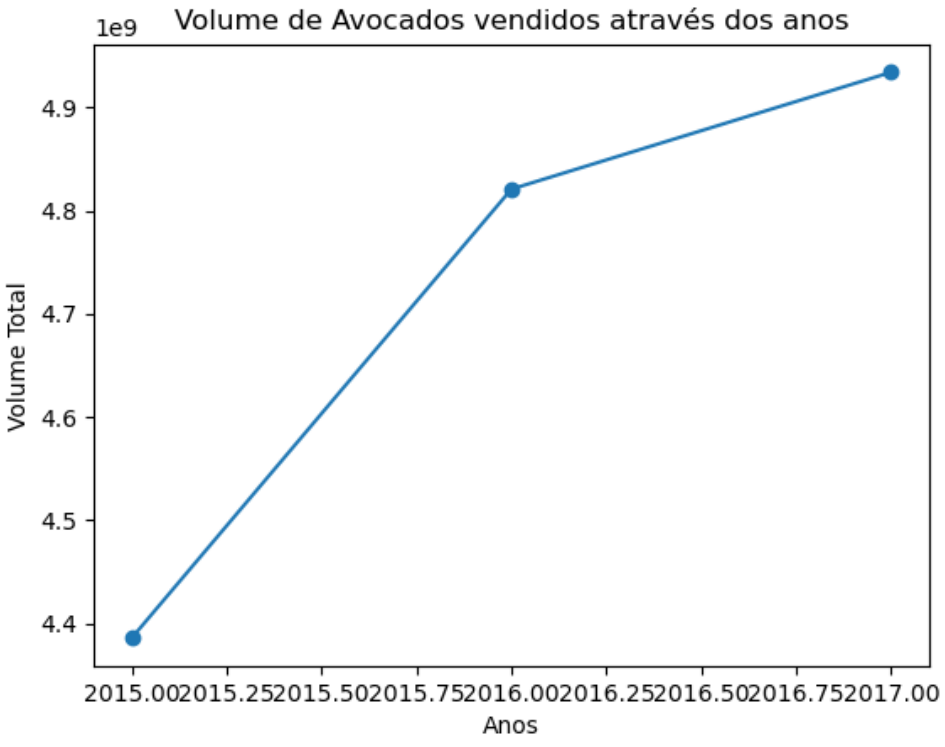
Durante a análise dos dados foram pensadas algumas perguntas que foram respondidas, como por exemplo: Qual a região que mais consome abacate?

Foram feitos vários testes então, o primeiro foi analisando o volume total de abacate vendido durante o correr do tempo.

O resultado que obtive não foi o planejado, pois a data final influenciou na demonstração, como o ano de 2018 está incompleto no dataset, então acabou sendo mostrado pelo gráfico que a melhor maneira de lidar com o ano de 2018 seria descartando ele do dataset.

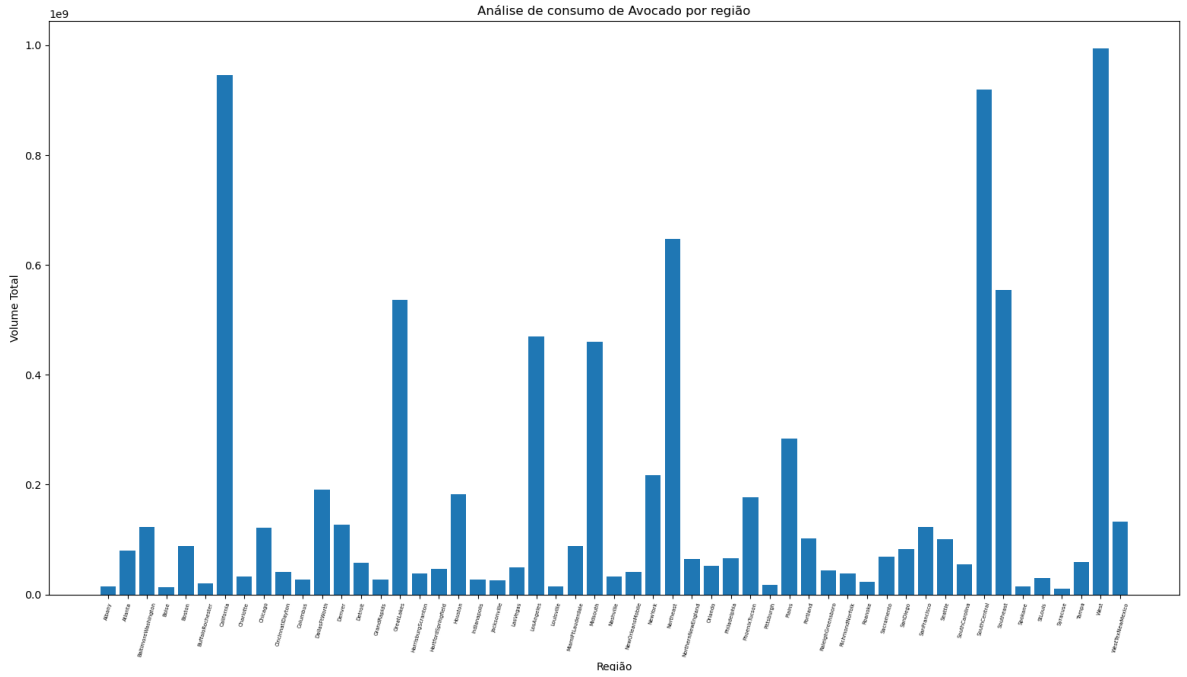


Podemos visualizar a queda no ano de 2018 e como pode acabar influenciando no resultado que queremos se não percebermos que o dataset para 2018 está incompleto.



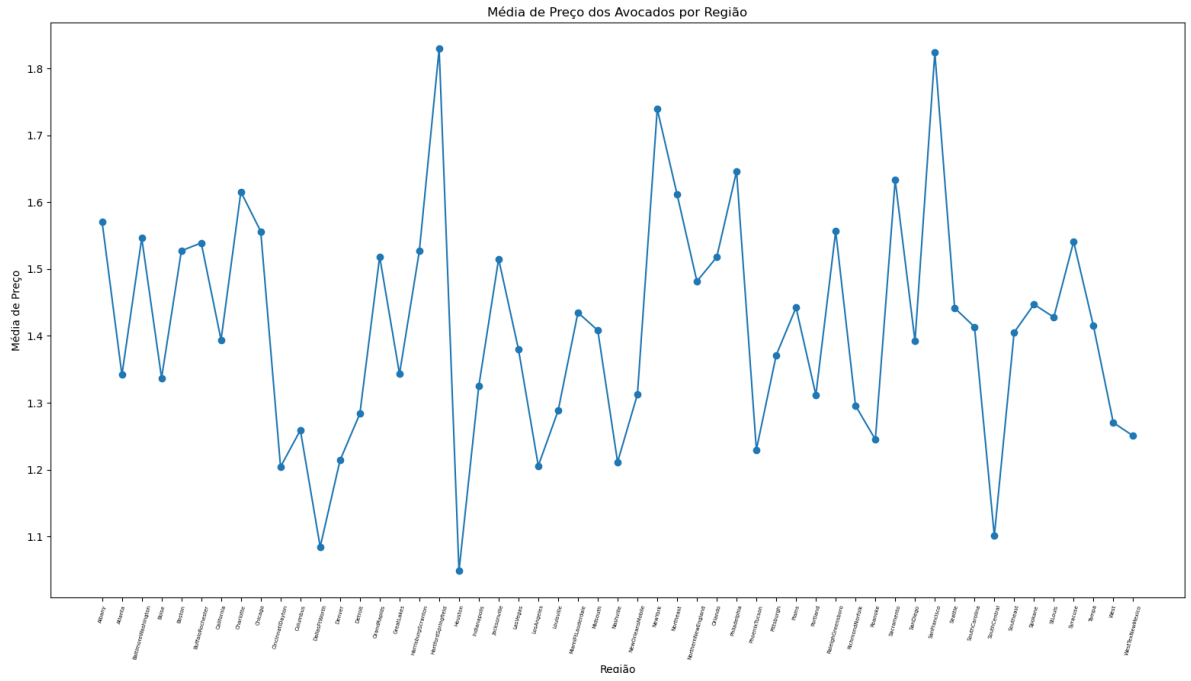
Acima podemos ter uma visualização mais nítida sobre o crescimento da venda de abacates durante os anos de 2015, 2016 e 2017.

Também há a visualização de consumo de abacate para cada região, analisando assim qual a região que mais consome abacate e assim a região que também menos consome.



Podemos analisar que a região que é maior consumidora de abacates é a WEST enquanto a menos consumista é a SYRACUSE.

E por último, realizei a visualização da média de preço para cada região:



Podemos ver que há uma alta variação entre as regiões, e novamente, a região mais cara para se obter um abacate é a HARTFORDSPRINGFIELD enquanto a mais barata sendo a HOUSTON.

5- Escolha de variáveis

Foram utilizadas todas as variáveis do dataset como X e como target foi utilizada a variável AVERAGEPRICE.

As colunas totais para o modelo foram:

'AVERAGEPRICE', 'TOTAL VOLUME', '4046', '4225', '4770', 'TOTAL BAGS',
'TYPE', 'REGION'

```
X = arquivo.drop(["AVERAGEPRICE"], axis=1)
y = arquivo["AVERAGEPRICE"]
```

6- Escolha do melhor modelo

Criei uma função que mostra o resultado de cada algorítmo que eu colocar e assim pode
me mostrar qual o melhor modelo que eu possa usar, com os dados de treino então o
resultado pode ser conferido logo abaixo.

```
def retorna_resultado_modelo(tipo_modelo_parametro):
    # REALIZANDO O FIT E PREDICT DO MODELO

    pred = tipo_modelo_parametro.fit(X_train, y_train).predict(X_test)

    # REALIZANDO TESTES DE DESEMPENHO DE MODELO

    """MEAN SQUARED ERROR"""
    mse = mean_squared_error(y_test, pred)

    """MEAN ABSOLUTE ERROR"""
    mae = mean_absolute_error(y_test, pred)

    """R2"""
    r2 = r2_score(y_test, pred)

    """MAX ERROR"""
    max_erro = max_error(y_test, pred)

    return print(f"MODELO {tipo_modelo_parametro}\nMSE: {mse}\nMAE: {mae}\nR2:
{r2}\n"
                 f"MAX ERROR: {max_erro}\n")


lista_modelos = [RandomForestRegressor(random_state=1), LinearRegression(),
Ridge(solver="auto", alpha=1.0),
                 DecisionTreeRegressor(max_depth=3, random_state=1)]

for tipo_modelo in lista_modelos:
    retorna_resultado_modelo(tipo_modelo)
```

O resultado de cada modelo pode ser observado abaixo:

MODELO RandomForestRegressor(random_state=1)
MSE: 3.25618306204298e-06
MAE: 4.5765510735496985e-05
R2: 0.9999803363383457
MAX ERROR: 0.11560000000000015

MODELO LinearRegression()
MSE: 4.589679316104591e-29
MAE: 5.461569180186566e-15

R2: 1.0
MAX ERROR: 3.064215547965432e-14

MODELO Ridge()
MSE: 3.643617372141119e-08
MAE: 0.00015379285645732526
R2: 0.9999997799667345
MAX ERROR: 0.0008633438717491337

MODELO DecisionTreeRegressor(max_depth=3, random_state=1)
MSE: 0.0053687595415499445
MAE: 0.05744863009440483
R2: 0.9675787665752208
MAX ERROR: 0.7503947368421042

Pelos resultados pude concluir então que o melhor modelo a ser utilizado na base real seria
o algoritmo de Ridge, com altos valores de R2, MAX ERROR e baixo
valor de MSE e MAE são também.

7- Treinamento do modelo

Feita a exportação do modelo preparado com a função pickle, assim podendo ser utilizado
no segundo arquivo.

```
modelo = Ridge(solver="auto", alpha=1.0).fit(X_train, y_train)

finalizado = "modelo_finalizado.sav"

pickle.dump(modelo, open(finalizado, "wb"))
```

8- Treinamento no arquivo de teste

```
modelo_carregado = pickle.load(open("modelo_finalizado.sav", "rb"))

# REALIZANDO O PREDICT DO ARQUIVO DE TESTE COM O MODELO IMPORTADO

X = arquivo_teste[["TOTAL VOLUME", "4046", "4225", "4770", "TOTAL BAGS", "TYPE", "REGION"]]
```

```
arquivo_teste["RESULTADO"] = modelo_carregado.predict(X)
```

O resultado pode ser conferido abaixo:

```
arquivo_teste["RESULTADO"]
Out[43]:
0       1.557677
1       1.123268
2       1.546622
3       1.037530
4       1.134518
         ...
8471    1.570979
8472    1.009387
8473    1.260352
8474    1.181496
8475    1.025325
Name: RESULTADO, Length: 8476, dtype: float64
```

9- Conclusão

Pude verificar um bom resultado com o modelo, a média dos valores se encaixam com cada região e valor médio estimado.