Product Demand Prediction with Machine Learnings – Guidelines

**Phase 4: Development Part 2**

**Continue building the product demand prediction model by:**

**Feature engineering**

**Model training**

**Evaluation.**

**Feature engineering**

**program**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

**Program;**

import numpy as np

import pandas as pd

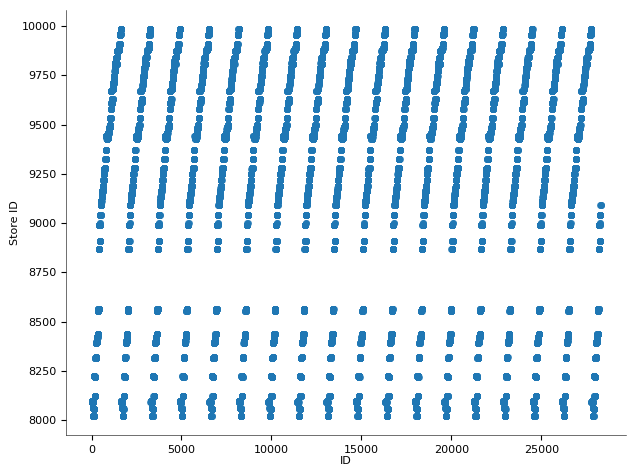
df=pd.read\_csv("/content/drive/MyDrive/PoductDemand.csv")

df

**output**

| **ID** | **Store ID** | **Total Price** | **Base Price** | **Units Sold** |
| --- | --- | --- | --- | --- |
| **0** | 1 | 8091 | 99.0375 | 111.8625 | 20 |
| **1** | 2 | 8091 | 99.0375 | 99.0375 | 28 |
| **2** | 3 | 8091 | 133.9500 | 133.9500 | 19 |
| **3** | 4 | 8091 | 133.9500 | 133.9500 | 44 |
| **4** | 5 | 8091 | 141.0750 | 141.0750 | 52 |
| **...** | ... | ... | ... | ... | ... |
| **150145** | 212638 | 9984 | 235.8375 | 235.8375 | 38 |
| **150146** | 212639 | 9984 | 235.8375 | 235.8375 | 30 |
| **150147** | 212642 | 9984 | 357.6750 | 483.7875 | 31 |
| **150148** | 212643 | 9984 | 141.7875 | 191.6625 | 12 |
| **150149** | 212644 | 9984 | 234.4125 | 234.4125 | 15 |
|  |  |  |  |  |  |

**Output**

****

**Model training**

**Program**

**from** **sklearn.feature\_extraction** **import** DictVectorizer

vec = DictVectorizer(sparse=**False**, dtype=int)

vec.fit\_transform(data)

**output**

array([[ 0, 1, 0, 850000, 4],

[ 1, 0, 0, 700000, 3],

[ 0, 0, 1, 650000, 3],

[ 1, 0, 0, 600000, 2]], dtype=int64)

**Program**

vec.get\_feature\_names()

**output**

['neighborhood=Fremont',

'neighborhood=Queen Anne',

'neighborhood=Wallingford',

'price',

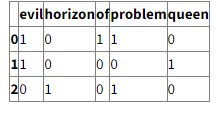
'rooms']

Program

**import** **pandas** **as** **pd**

pd.DataFrame(X.toarray(), columns=vec.get\_feature\_names())

output



Program

|  |
| --- |
| from sklearn.linear\_model import LinearRegression |
|  |  |
|  | predictor = LinearRegression(n\_jobs=-1) |
|  | predictor.fit(X=TRAIN\_INPUT, y=TRAIN\_OUTPUT) |

Output

Outcome : [ 140.]

Coefficients : [ 1. 2. 3.]

**Evaluation**

Program

# make a single prediction with the model

from sklearn.linear\_model import LogisticRegression

from sklearn.datasets import make\_blobs

# create the inputs and outputs

X, y = make\_blobs(n\_samples=1000, centers=2, n\_features=2, random\_state=2)

# define model

model = LogisticRegression(solver='lbfgs')

# fit model

model.fit(X, y)

# make predictions on the entire training dataset

yhat = model.predict(X)

# connect predictions with outputs

for i in range(10):

print(X[i], yhat[i]

output

[ 1.23839154 -2.8475005 ] 1

[-1.25884111 -8.57055785] 0

[ -0.86599821 -10.50446358] 0

[ 0.59831673 -1.06451727] 1

[ 2.12309797 -1.41131072] 1

[-1.53722693 -9.61845366] 0

[ 0.92194131 -0.68709327] 1

[-1.31478732 -8.78528161] 0

[ 1.57989896 -1.462412 ] 1

[ 1.36989667 -1.3964704 ] 1

Program

from matplotlib import pyplot as plt

from sklearn import svm

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import r2\_score, mean\_squared\_error, mean\_absolute\_error, median\_absolute\_error, explained\_variance\_score, max\_error

|  |  |
| --- | --- |
|  | test = df.iloc[-52:] |
|  | df = df.iloc[:-52] |

X = df.drop('sales', axis=1)

y = df['sales']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

clf = svm.SVR(C=1, kernel='linear', degree=8, gamma='scale', coef0=10)

clf.fit(X\_train, y\_train)

predictions = clf.predict(X\_test)

print(f'Model fit results:\n'

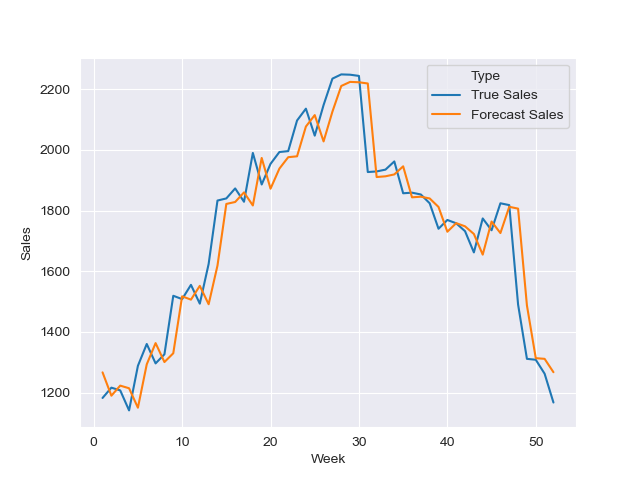
f'r2\_score {r2\_score(y\_test, predictions)} \t MSE {mean\_squared\_error(y\_test, predictions)}'

f'\tEVS {explained\_variance\_score(y\_test, predictions)} \n MAE {mean\_absolute\_error(y\_test, predictions)}'

f'\tMAD {median\_absolute\_error(y\_test, predictions)}\t ME {max\_error(y\_test, predictions)}')

output

###OUTPUT###  
Model fit results:  
r2\_score 0.9071953443448584 MSE 6553.674543344077   
EVS 0.9175800366290838 MAE 58.80295451823111   
MAD 37.648574124556035 ME 304.51308147895793



Conculsion

Remember to split your dataset into training and testing sets to evaluate the model's performance on unseen data. Hyperparameter tuning may be necessary to optimize the model's performance. Iterate through feature engineering, model training, and evaluation until you achieve satisfactory results.