**PRODUCT DEMAND PREDICTION WITH MACHINE**

**LEARNING [APPLIED DATA SCIENCE]**

**PHASE IV PROJECT: FEATURE ENGINEERING, MODEL TRAINING AND EVALUATION**

**TABLE OF CONTENTS:**

1. **INTRODUCTION**
2. **PROBLEM DEFINITION**
3. **FEATURE ENGINEERING**
4. **MODEL TRAINING**
5. **EVALUTION**
6. **CONCLUSION**
7. **REFERENCES**

**INTODUCTION:**

The problem we have taken is prediction of demand for products using machine learning. We know that the demand for products varies with time and place in real life. With demand the product price increases and decreases. We have to create a machine learning model that forecasts product demand based on historical sales data. In this phase’s project we are going to do feature engineering and train the model and finally evaluate the model’s efficiency.

**PROBLEM DEFINITON:**

The problem of predicting demand for a new product based on its characteristics and description is critical for various industrial enterprises, wholesale and retail trade and, especially, for modern highly competitive sector of air transportation, since solving this problem will optimize production, management and logistics in order to maximize profits and minimize costs.

**FEATURE ENGINEERING:**

**Date and time feature:**

The date and time feature is used to observe the date/ time of each observation. We use pandas here and create a data frame of the columns in the dataset for getting the head off values.

These are components of the time step itself for each observation

dataframe['units sold'] = [data.index[i].month for i in range(len(data))]

dataframe['base price'] = [data.index[i].day for i in range(len(data))]

print(dataframe.head(5))

**Rolling windows feature:**

This rolling windows feature is used to find summaries of data. these are a summary of values over a fixed window of prior time steps.

Here we used to find the mean, minimum and maximum of the data.

import pandas as pd

dataframe = dataframe.sort\_values(by='id')

window\_size =5

df['rolling\_mean']=df['demand'].rolling(window=window\_size).mean()

df['rolling\_min']=df['demand'].rolling(window=window\_size).min()

df['rolling\_max']=df['demand'].rolling(window=window\_size).max()

df = df.dropna()

print(df.head())

**lag features:**

These are values at prior time steps.

from pandas import DataFrame

from pandas import concat read\_csv('daily-min-temperatures.csv', header=0, index\_col=0)

temps = DataFrame(data.values)

dataframe=concat([temps.shift(3),temps.shift(2),temps.shift(1), temps], axis=1)

dataframe.columns = ['t-3', 't-2', 't-1', 't+1']

print(dataframe.head(5))

**MODEL TRAINING:**

import pandas as pd

import numpy as np

from sklearn import tree

from sklearn import datasets

from sklearn.datasets import load\_iris

from sklearn.tree import DecisionTreeClassifier

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

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.metrics import precision\_score,

recall\_score, f1\_score, accuracy\_score

data = pd.read\_csv('demand\_data.csv')

X = data.drop(columns=["Units Sold"])

y = data["Units Sold"]

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

model = LinearRegression()

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

y\_pred = model.predict(X\_test)

**EVALUATION:**

from sklearn.metrics import precision\_score,

recall\_score, f1\_score, accuracy\_score

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=20, test\_size=0.20)

tree = DecisionTreeClassifier()

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

y\_pred = tree.predict(X\_test)

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Precision:",precision\_score(y\_test,y\_pred, average="weighted"))

print('Recall:'recall\_score(y\_test,y\_pred average="weighted"))

print('F1 score:', f1\_score(y\_test, y\_pred, average="weighted"))

confusion\_matrix = metrics.confusion\_matrix(y\_test, y\_pred)

cm\_display=metrics.ConfusionMatrixDisplay( confusion\_matrix=confusion\_matrix, display\_labels=[0, 1, 2])

cm\_display.plot()

plt.show()

import numpy as np

from sklearn .metrics import roc\_auc\_score

y\_true = [1, 0, 0, 1]

y\_pred = [1, 0, 0.9, 0.2]

auc = np.round(roc\_auc\_score(y\_true, y\_pred), 3)

print("Auc", (auc))

# importing the libraries

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_absolute\_error,

mean\_squared\_error, mean\_absolute\_percentage\_error

df = pd.read\_csv('weather.csv')

X = df.iloc[:, 2].values

Y = df.iloc[:, 3].values

X\_train, X\_test,

Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.20, random\_state=0)

X\_train = X\_train.reshape(-1, 1)

X\_test = X\_test.reshape(-1, 1)

regression = LinearRegression()

regression.fit(X\_train, Y\_train)

Y\_pred = regression.predict(X\_test)

mae = mean\_absolute\_error(y\_true=Y\_test, y\_pred=Y\_pred)

print("Mean Absolute Error", mae)

mse = mean\_squared\_error(y\_true=Y\_test, y\_pred=Y\_pred)

print("Mean Square Error", mse)

rmse = mean\_squared\_error(y\_true=Y\_test, y\_pred=Y\_pred,

squared=False)

print("Root Mean Square Error", rmse)

mape = mean\_absolute\_percentage\_error(Y\_test, Y\_pred,

sample\_weight=None, multioutput='uniform\_average')

print("Mean Absolute Percentage Error", mape)

**CONCLUSION:**

Thus we have developed a model with feature engineering, trained a model and evaluated it.

**REFERENCES:**

<https://www.geeksforgeeks.org/machine-learning-model-evaluation/>

<https://machinelearningmastery.com/basic-feature-engineering-time-series-data-python/>