PRODUCT DEMAND PREDICTION WITH MACHINE LEARNINGS:

912421104002 : ABIRAMI.M

PHASE-4 PROJECT SUBMISSION DOCUMENT

PROJECT TITLE: PRODUCT DEMAND PERDICTION

PHASE-4: DEVELOPMENT PART 2

TOPIC: TO CONTINUE BUILDING THE PRODUCT DEMAND PREDICTION MODEL BY PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE ENGINEERING,MODEL TRAINING,EVALUATION,ETC.

PRODUCT DEMAND PREDICTION:

INTRODUCTION:

- ❖ The process of Building a product demand prediction model involves the systematic process of creating a mathematical or computational model that forecasts the future demand for a product or a set of products. This process aims to leverage historical data, relevant features, and statistical or machine learning techniques to make accurate predictions about future demand.

- ❖ The process of building the product demand prediction model can vary depending on the type and complexity of the data, the business objectives, and the available tools.

- ❖ Building a product demand prediction model involves the systematic process of creating a mathematical or computational model that forecasts the future demand for a product or a set of products. This process aims to leverage historical data, relevant features, and statistical or machine learning techniques to make accurate predictions about future demand. The main steps involved in this process include:

- ❖ Feature Engineering is the process of Create or extract relevant features from the data that can help the model make accurate predictions. These features may include time-related variables (seasonality and trends), promotional events, historical demand, and external factors that impact demand.

- ❖ Model Training is the process of train the selected model using the training data. During training, the model learns from historical patterns and relationships in the data.

- ❖ Model Evaluation is assess the model's performance using the validation dataset. Common evaluation metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). The model may need fine-tuning based on the evaluation results.

<u>Given Dataset:</u>

ID	Store ID	Total Price	Base Price	Units Sold
1	8091	99.0375	111.8625	20
2	8091	99.0375	99.0375	28
3	8091	133.95	133.95	19
4	8091	133.95	133.95	44
5	8091	141.075	141.075	52
9	8091	227.2875	227.2875	18
10	8091	327.0375	327.0375	47
13	8091	210.9	210.9	50

14	8091	190.2375	234.4125	82
17	8095	99.0375	99.0375	99
18	8095	97.6125	97.6125	120
19	8095	98.325	98.325	40
22	8095	133.2375	133.2375	68
23	8095	133.95	133.95	87
24	8095	139.65	139.65	186
27	8095	236.55	280.0125	54
28	8095	214.4625	214.4625	74
29	8095	266.475	296.4	102
30	8095	173.85	192.375	214
31	8095	205.9125	205.9125	28
32	8095	205.9125	205.9125	7
33	8095	248.6625	248.6625	48
34	8095	200.925	200.925	78
35	8095	190.2375	240.825	57
37	8095	427.5	448.1625	50
38	8095	429.6375	458.1375	62
39	8095	177.4125	177.4125	22
42	8094	87.6375	87.6375	109
43	8094	88.35	88.35	133
44	8094	85.5	85.5	11
45	8094	128.25	180.975	9
47	8094	127.5375	127.5375	19
48	8094	123.975	123.975	33
49	8094	139.65	164.5875	49
50	8094	235.8375	235.8375	32
51	8094	234.4125	234.4125	47
52	8094	235.125	235.125	27
53	8094	227.2875	227.2875	69
54	8094	312.7875	312.7875	49
55	8094	210.9	210.9	60
56	8094	177.4125	177.4125	27
57	8094	177.4125	177.4125	33
58	8094	240.825	240.825	18
59	8094	213.0375	213.0375	72
60	8094	190.95	213.0375	81
61	8094	426.7875	448.1625	11
62	8094	426.7875	448.875	13
63	8094	426.7875	448.1625	28
65	8094	170.2875	170.2875	16

Overview of the process:

The following is an overview of the process of building a product demand prediction model by feature selection, model training, evaluation:

1. Define the Problem:

- Clearly define the problem you want to solve. What product or products are you trying to predict demand for? What are your specific goals and objectives?

2. Data Collection:

- Gather historical data related to the product's sales, including sales volume, price, and any other relevant variables. Additional data sources may include marketing activities, seasonality, economic indicators, and external factors.

3. Data Preprocessing:

- Clean and preprocess the collected data. This may involve handling missing data, outliers, and ensuring data consistency.

4. Feature Engineering:

- Create meaningful features from the raw data. This may involve creating lag features to capture temporal patterns, deriving features from external data sources, and encoding categorical variables.

5. Data Splitting:

- Split your dataset into training, validation, and testing sets. The training set is used to train the model, the validation set helps fine-tune model parameters, and the testing set is used to evaluate the model's performance.

6. Model Selection:

- Choose an appropriate modeling technique for demand prediction. Common approaches include time series forecasting methods (e.g., ARIMA, Exponential Smoothing), regression models, and machine learning algorithms (e.g., linear regression, decision trees, neural networks).

7. Model Training:

- Train your chosen model on the training dataset. This involves optimizing model parameters to minimize the prediction error.

8. Model Evaluation:

- Assess the model's performance using the validation dataset. Common evaluation metrics for demand prediction include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared.

9. Hyperparameter Tuning:

- Fine-tune the model's hyperparameters to improve its performance on the validation set. Techniques like grid search or random search can be used for this purpose.

10. Model Validation:

- Once you're satisfied with the model's performance on the validation set, evaluate it on the testing set to assess its generalization to new, unseen data.

11. Deployment:

- Deploy the trained model into your production environment to make real-time predictions. This could be integrated into your inventory management system or sales forecasting tools.

12. Monitoring and Maintenance:

- Continuously monitor the model's performance in the production environment. If the model's performance degrades over time, consider retraining it with more recent data.

13. Feedback Loop:

- Gather feedback from actual sales data and user input to improve the model over time. Use this feedback to iterate and refine your demand prediction model.

14. Documentation:

- Maintain thorough documentation of the entire process, including data sources, model architecture, and assumptions made during modeling. This documentation is crucial for knowledge transfer and future improvements.

Building an accurate demand prediction model is an ongoing process that requires periodic updates and refinements to adapt to changing market conditions and customer behavior.

Procedure:

FEATURE ENGINEERING:

Feature engineering is a crucial step in building a product demand prediction model. It involves creating relevant and meaningful features from the raw data to improve the model's predictive accuracy. Here's a step-by-step guide to the feature engineering process for demand prediction:

1. Understanding the Data:

- Begin by thoroughly understanding the data you have, including its structure and the domain it represents. This will help you make informed decisions when engineering features.

2. Domain Knowledge:

- Leverage domain expertise to identify potential features that could impact product demand. Speak to subject matter experts or conduct a literature review to gather insights.

3. Feature Selection:

- Decide which features you will use in your model. Select those that are relevant to demand prediction and have a reasonable expectation of influencing demand. Features could include:

  - Historical sales data

- Price and discount information
- Marketing campaigns and promotions
- Seasonal information
- Economic indicators (e.g., GDP, inflation)
- External factors (e.g., weather data)

4. Lag Features:

- Create lag features to capture temporal dependencies. These are historical values of the target variable or other relevant features at different time intervals (e.g., daily, weekly, monthly). Lag features help the model capture trends and seasonality.

5. Moving Averages and Aggregations:

- Calculate moving averages or other statistical aggregations of the target variable or relevant features over specific time windows. This can help capture trends and smoothing effects.

6. Categorical Variable Encoding:

- If your data includes categorical variables (e.g., product categories, store locations), you need to encode them. Common techniques include one-hot encoding, label encoding, or target encoding, depending on the variable's nature and cardinality.

7. Feature Scaling:

- Normalize or scale your features if necessary. This ensures that features with different scales contribute equally to the model's predictions. Common methods include Min-Max scaling or z-score normalization.

8. External Data Integration:

- Incorporate external data sources that might impact product demand. For example, integrating weather data can be important for predicting demand for seasonal products.

9. Text Data Processing:

- If you have text data (e.g., customer reviews, product descriptions), you can use natural language processing techniques to extract relevant information. This might include sentiment analysis or keyword extraction.

10. Feature Interactions:

- Create new features that represent interactions between existing features. For example, you can multiply sales with marketing budget to capture the interaction effect.

11. Time-Related Features:

- Introduce time-related features such as day of the week, month, or holiday indicators. These can help capture day-of-week or seasonality effects.

12. Dimensionality Reduction:

- If your dataset has a large number of features, consider dimensionality reduction techniques like Principal Component

Analysis (PCA) to reduce the number of features while preserving important information.

13. Regularization Features:

- In some cases, you may create regularization features to penalize extreme values or trends that are not typical.

14. Feature Importance Analysis:

- Use feature importance techniques (e.g., feature importance scores from tree-based models) to identify which features have the most influence on the model's predictions. This can help refine feature selection.

15. Cross-Validation:

- When engineering features, ensure you use cross-validation to assess their impact on model performance and prevent overfitting.

16. Iterate:

- Feature engineering is often an iterative process. Keep refining your feature set based on the model's performance and domain knowledge.

Regularly reevaluate the feature engineering process as new data becomes available or business conditions change.

EXAMPLE PROGRAM CODE:

```
In [43]: from sklearn.feature_selection import SelectKBest
         from sklearn.feature_selection import chi2
         bestfeatures = SelectKBest(score_func=chi2, k=10)
         fit = bestfeatures.fit(x,y)
         dfscores = pd.DataFrame(fit.scores_)
         dfcolumns = pd.DataFrame(x.columns)
         featureScores = pd.concat([dfcolumns,dfscores],axis=1)
         featureScores.columns = ['Specs','Score']
         featureScores
```

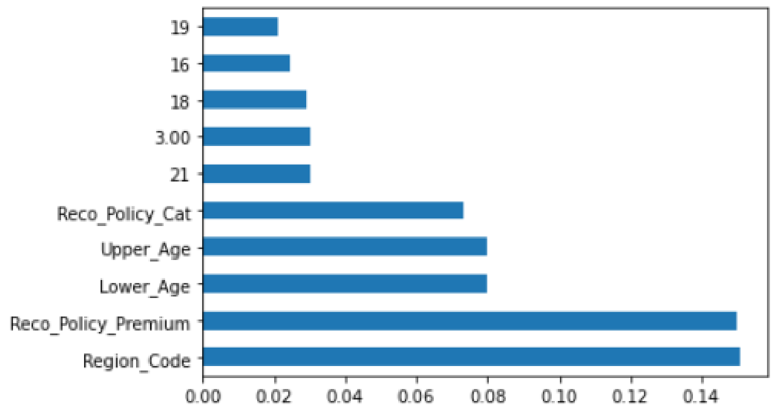
```
In [47]: featureScores
```

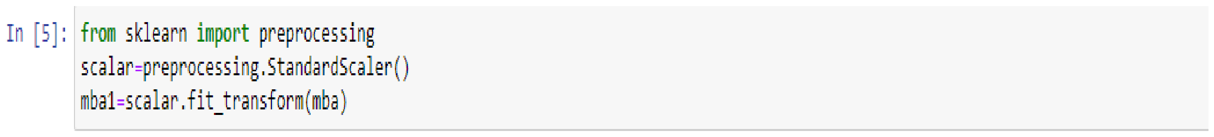
Out[47]:

	Specs	Score
0	City_Code	0.122643
1	Region_Code	74.805013
2	Accomodation_Type	0.756115
3	Reco_Insurance_Type	3.965972
4	Upper_Age	2.612166
5	Lower_Age	1.572930
6	Is_Spouse	0.632296
7	Health Indicator	0.453390
8	Holding_Policy_Duration	7.662646
9	Holding_Policy_Type	0.836189
10	Reco_Policy_Cat	1894.032997
11	Reco_Policy_Premium	9575.065324
12	diff_age	0.039347

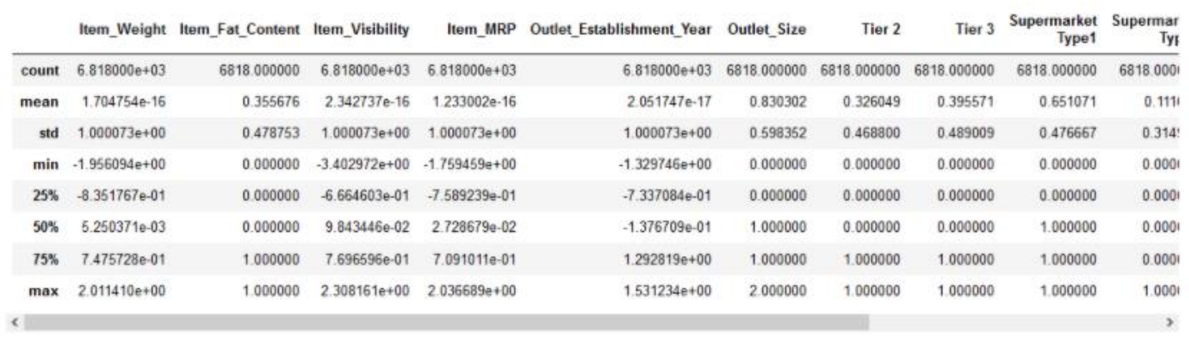
```
In [189]: from sklearn.ensemble import ExtraTreesClassifier
          import matplotlib.pyplot as plt
          model = ExtraTreesClassifier()
          model.fit(x,y)
          print(model.feature_importances_)

Out[189]: ExtraTreesClassifier()
```

```
In [192]: feat_importances = pd.Series(model.feature_importances_, index=x.columns)
          feat_importances.nlargest(10).plot(kind='barh')
          plt.show()
```

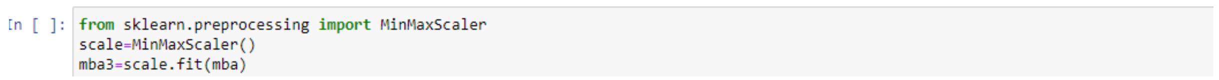




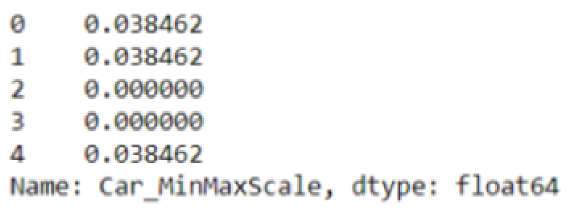
```
In [5]: from sklearn import preprocessing
        scalar=preprocessing.StandardScaler()
        mba1=scalar.fit_transform(mba)
```



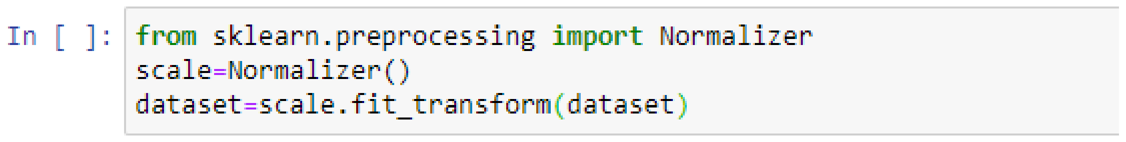
	Item_Weight	Item_Fat_Content	Item_Visibility	Item_MRP	Outlet_Establishment_Year	Outlet_Size	Tier 2	Tier 3	Supermarket Type1	Supermar Typ
count	6.818000e+03	6818.000000	6.818000e+03	6.818000e+03	6.818000e+03	6818.000000	6818.000000	6818.000000	6818.000000	6818.000
mean	1.704754e-16	0.355676	2.342737e-16	1.233002e-16	2.051747e-17	0.830302	0.326049	0.395571	0.651071	0.111
std	1.000073e+00	0.478753	1.000073e+00	1.000073e+00	1.000073e+00	0.598352	0.468800	0.489009	0.476667	0.314
min	-1.956094e+00	0.000000	-3.402972e+00	-1.759459e+00	-1.329746e+00	0.000000	0.000000	0.000000	0.000000	0.000
25%	-8.351767e-01	0.000000	-6.664603e-01	-7.589239e-01	-7.337084e-01	0.000000	0.000000	0.000000	0.000000	0.000
50%	5.250371e-03	0.000000	9.843446e-02	2.728679e-02	-1.376709e-01	1.000000	0.000000	0.000000	1.000000	0.000
75%	7.475728e-01	1.000000	7.696596e-01	7.091011e-01	1.292819e+00	1.000000	1.000000	1.000000	1.000000	0.000
max	2.011410e+00	1.000000	2.308161e+00	2.036689e+00	1.531234e+00	2.000000	1.000000	1.000000	1.000000	1.000



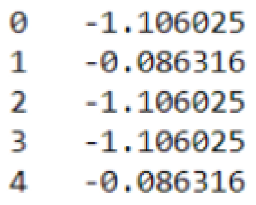
```
In [ ]: from sklearn.preprocessing import MinMaxScaler
        scale=MinMaxScaler()
        mba3=scale.fit(mba)
```



```
0    0.038462
1    0.038462
2    0.000000
3    0.000000
4    0.038462
Name: Car_MinMaxScale, dtype: float64
```



```
In [ ]: from sklearn.preprocessing import Normalizer
        scale=Normalizer()
        dataset=scale.fit_transform(dataset)
```



```
0   -1.106025
1   -0.086316
2   -1.106025
3   -1.106025
4   -0.086316
```

MODEL TRAINING:

The model training process for building a product demand prediction model involves preparing the data, selecting an appropriate modeling technique, training the model, and evaluating its performance. Here is a step-by-step guide for the model training process:

1. Data Preprocessing:

- Before training your model, preprocess the data to ensure it's in a suitable format for modeling. Common preprocessing steps include handling missing data, scaling or normalizing features, encoding categorical variables, and splitting the data into training and validation sets.

2. Select an Appropriate Model:

- Choose a modeling technique that is suitable for your specific demand prediction task. Common models used for demand prediction include:

  - Time Series Models: such as ARIMA, Exponential Smoothing, or Prophet for capturing time-dependent patterns.

  - Regression Models: like linear regression, decision trees, random forests, or gradient boosting for capturing linear and nonlinear relationships between features and demand.

  - Machine Learning Models: such as neural networks (e.g., deep learning), support vector machines, or k-nearest neighbors, which can capture complex patterns and relationships in the data.

3. Train the Model:

- Train the selected model on your training data. The steps involved in training depend on the type of model:

  - Time Series Models: You would typically estimate model parameters using historical demand data.

  - Regression Models: Use an optimization algorithm to find the best coefficients that minimize the prediction error (e.g., mean squared error).

  - Machine Learning Models: The training process involves adjusting the model's internal parameters to minimize a loss function, usually involving gradient descent or variations thereof.

4. Hyperparameter Tuning:

- Fine-tune the hyperparameters of your model to optimize its performance. You can use techniques like grid search, random search, or Bayesian optimization to find the best hyperparameters. This step is especially important for machine learning models.

5. Cross-Validation:

- Use cross-validation, such as k-fold cross-validation, to assess how well your model generalizes to new data and to estimate its performance more accurately. This helps prevent overfitting.

6. Model Evaluation:

- Assess the model's performance using appropriate evaluation metrics. Common metrics for demand prediction include Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared. Evaluate the model on both the training and validation datasets.

7. Feature Importance:

- For machine learning models, determine the importance of individual features in making predictions. This information can help in feature selection and understanding the drivers of demand.

8. Model Interpretability:

- For complex models like neural networks, consider techniques for making the model more interpretable, such as feature importance plots or SHAP (SHapley Additive exPlanations) values.

9. Model Selection:

- Compare the performance of different models and choose the one that performs best on the validation data. Consider factors like interpretability, computational resources, and ease of implementation.

10. Final Model Training:

- Train the selected model on the entire training dataset, using the optimal hyperparameters, to create the final model that will be used for making predictions.

11. Save the Model:

- Save the trained model to a file or database so that it can be easily loaded and used for future predictions without having to retrain it.

12. Documentation:

- Maintain documentation that includes details of the selected model, its hyperparameters, and its performance on the training and validation datasets. This documentation is essential for model maintenance and future improvements.

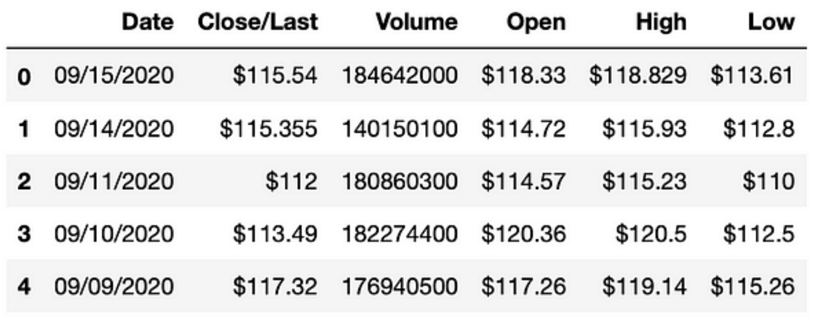
It's important to continually monitor and update the model to ensure that it remains accurate and relevant for demand prediction.

EXAMPLE PROGRAM CODE:

```
import pandas as pd
import numpy as npimport matplotlib.pyplot as plt
%matplotlib inline
from matplotlib.pylab import rcParams
rcParams['figure.figsize']=20,10from keras.models import Sequential
from keras.layers import LSTM,Dropout,Densefrom
sklearn.preprocessing import MinMaxScaler

import pandas as pddf = pd.read_csv('aapl_stock_1yr.csv')
```

```
df.head()
```

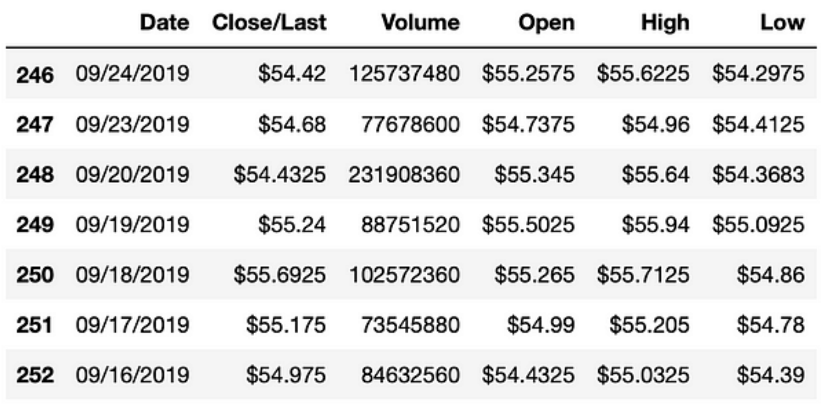
OUTPUT:



	Date	Close/Last	Volume	Open	High	Low
0	09/15/2020	$115.54	184642000	$118.33	$118.829	$113.61
1	09/14/2020	$115.355	140150100	$114.72	$115.93	$112.8
2	09/11/2020	$112	180860300	$114.57	$115.23	$110
3	09/10/2020	$113.49	182274400	$120.36	$120.5	$112.5
4	09/09/2020	$117.32	176940500	$117.26	$119.14	$115.26

```
df.tail()
```

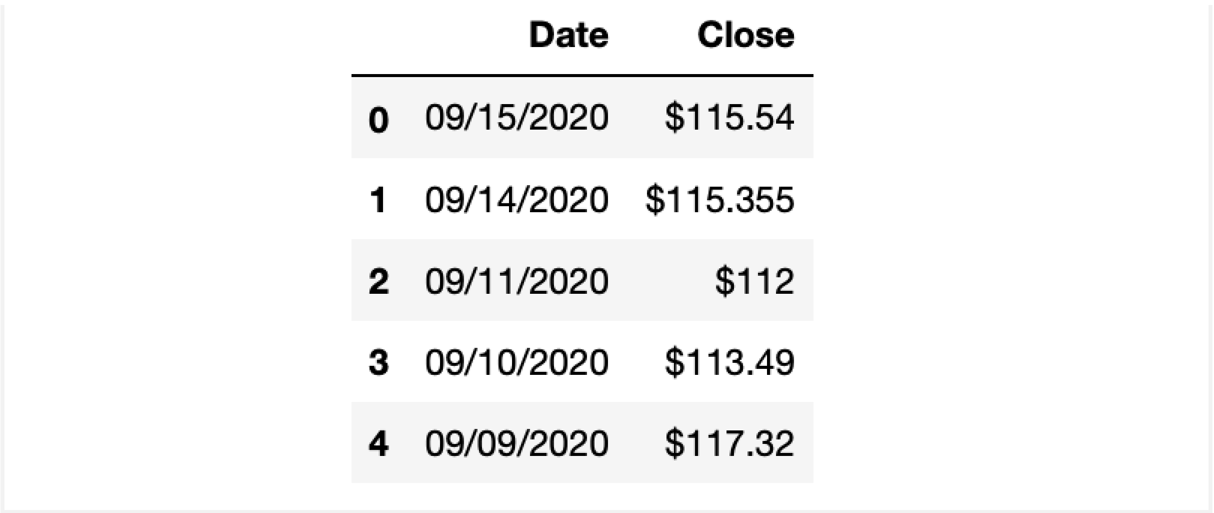
OUTPUT:



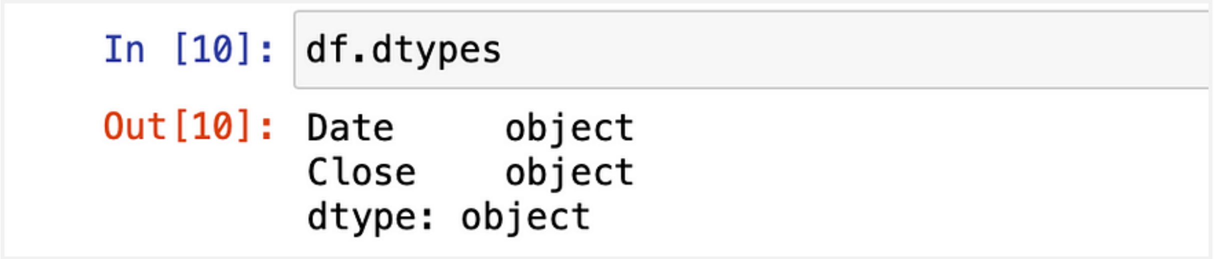
	Date	Close/Last	Volume	Open	High	Low
246	09/24/2019	$54.42	125737480	$55.2575	$55.6225	$54.2975
247	09/23/2019	$54.68	77678600	$54.7375	$54.96	$54.4125
248	09/20/2019	$54.4325	231908360	$55.345	$55.64	$54.3683
249	09/19/2019	$55.24	88751520	$55.5025	$55.94	$55.0925
250	09/18/2019	$55.6925	102572360	$55.265	$55.7125	$54.86
251	09/17/2019	$55.175	73545880	$54.99	$55.205	$54.78
252	09/16/2019	$54.975	84632560	$54.4325	$55.0325	$54.39

```
df = df[['Date', 'Close']]df.head()
```

OUTPUT:



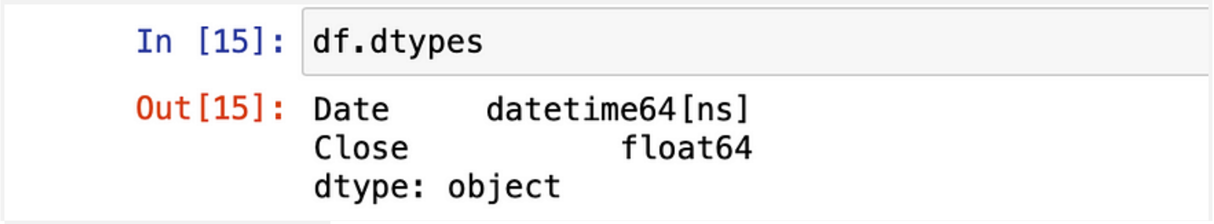
	Date	Close
0	09/15/2020	$115.54
1	09/14/2020	$115.355
2	09/11/2020	$112
3	09/10/2020	$113.49
4	09/09/2020	$117.32



```
In [10]: df.dtypes

Out[10]: Date      object
         Close     object
         dtype: object
```

df = df.replace({'\$':''}, regex = True)

df = df.astype({"Close": float})df["Date"] = pd.to_datetime(df.Date, format="%m/%d/%Y")df.dtypes

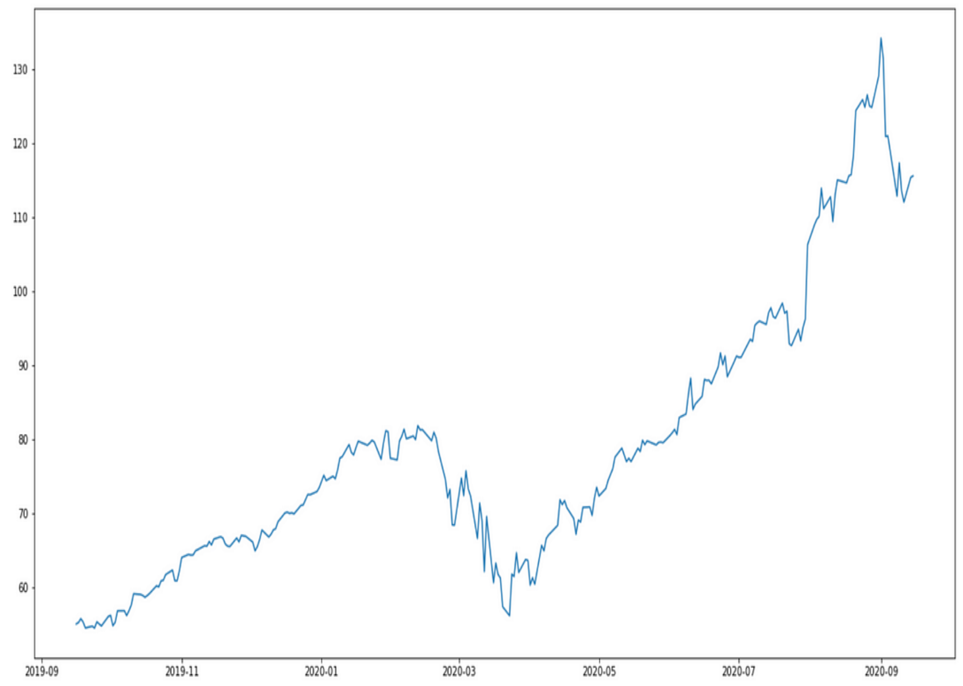


```
In [15]: df.dtypes

Out[15]: Date      datetime64[ns]
         Close            float64
         dtype: object
```

df.index = df['Date']

plt.plot(df["Close"],label='Close Price history')

```
In [19]: plt.plot(df["Close"],label='AAPL Close Price history')
Out[19]: [<matplotlib.lines.Line2D at 0x14b306c50>]
```



MODEL EVALUATION:

The model evaluation process is a critical step in building a product demand prediction model. It involves assessing the model's performance to determine how well it can accurately predict future product demand. Here's a step-by-step guide for the model evaluation process:

1. Data Splitting:

- Start by splitting your dataset into distinct subsets: a training set, a validation set, and a testing set. A common split might be 70% for training, 15% for validation, and 15% for testing. The training set is used to train the model, the validation set helps fine-tune hyperparameters, and the testing set is reserved for final evaluation.

2. Choose Evaluation Metrics:

- Select appropriate evaluation metrics that are relevant to your demand prediction task. Common metrics include:

  - **Mean Absolute Error (MAE)**: Measures the average absolute difference between predicted and actual demand.

  - **Mean Squared Error (MSE)**: Measures the average squared difference between predicted and actual demand, giving more weight to large errors.

  - **Root Mean Squared Error (RMSE)**: The square root of MSE, providing a measure in the same units as the target variable.

  - **R-squared (R^2)**: Indicates the proportion of variance in the target variable explained by the model. A higher R-squared value is generally better.

3. Model Evaluation on Validation Set:

- Assess your model's performance on the validation set using the chosen evaluation metrics. This is an essential step for fine-tuning hyperparameters and making adjustments to the model if needed.

4. Hyperparameter Tuning:

- If your model's performance on the validation set is not satisfactory, perform hyperparameter tuning. Adjust the model's hyperparameters and repeat the training and evaluation steps until you achieve the desired performance.

5. Cross-Validation:

- To obtain a more robust estimate of your model's performance and to prevent overfitting, you can use cross-validation techniques such as k-fold cross-validation. This involves splitting the data into multiple folds and training/evaluating the model multiple times.

6. Final Model Selection:

- After fine-tuning and optimizing your model on the validation set, select the best-performing model to move forward. You may choose the model with the lowest error or the highest R-squared, depending on your specific goals.

7. Model Evaluation on the Testing Set:

- Once you have chosen your final model, evaluate its performance on the testing set. This provides an unbiased assessment of how well the model will perform on unseen data.

8. Visualizations:

- Create visualizations such as time series plots, prediction vs. actual demand charts, and residual plots to gain insights into your model's behavior and errors.

9. Interpretability:

- If applicable, assess the model's interpretability. Depending on the model type, consider methods such as feature importance analysis or SHAP (Shapley Additive Explanations) values to understand which features drive predictions.

10. Benchmarking:

- Compare your model's performance to a simple baseline model (e.g., using historical average demand) to determine how much improvement your model provides.

11. Documentation and Reporting:

- Document the results of your model evaluation, including key metrics, findings, and any insights gained. This documentation is important for knowledge sharing and future reference.

12. Regular Monitoring and Reevaluation:

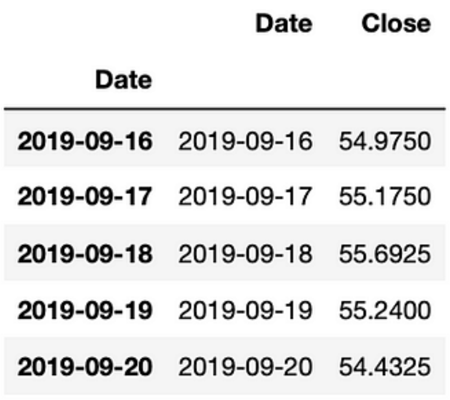
- After deploying the model in a production environment, continually monitor its performance. Reevaluate and update the model as needed with new data to ensure it remains accurate over time.

It's important to maintain a robust evaluation framework to ensure the model remains effective in practice.

EXAMPLE PROGRAM CODE:

```
df = df.sort_index(ascending=True,axis=0)data =
pd.DataFrame(index=range(0,len(df)),columns=['Date','Close'])for i in
range(0,len(data)):
   data["Date"][i]=df['Date'][i]
   data["Close"][i]=df["Close"][i]data.head()
```

OUTPUT:



	Date	Close
Date		
2019-09-16	2019-09-16	54.9750
2019-09-17	2019-09-17	55.1750
2019-09-18	2019-09-18	55.6925
2019-09-19	2019-09-19	55.2400
2019-09-20	2019-09-20	54.4325

Min-Max Scaler

```
scaler=MinMaxScaler(feature_range=(0,1))data.index=data.Date
data.drop(“Date”,axis=1,inplace=True)final_data = data.values
train_data=final_data[0:200,:]
valid_data=final_data[200:,:]scaler=MinMaxScaler(feature_range=(0,1))
scaled_data=scaler.fit_transform(final_data)
x_train_data,y_train_data=[],[]
for i in range(60,len(train_data)):
   x_train_data.append(scaled_data[i-60:i,0])
   y_train_data.append(scaled_data[i,0])
```

LSTM Model

```
lstm_model=Sequential()
lstm_model.add(LSTM(units=50,return_sequences=True,input_shape=(
np.shape(x_train_data)[1],1)))
lstm_model.add(LSTM(units=50))
lstm_model.add(Dense(1))model_data=data[len(data)-len(valid_data)-
60:].values
model_data=model_data.reshape(-1,1)
model_data=scaler.transform(model_data)
```

Train and Test Data

```
lstm_model.compile(loss='mean_squared_error',optimizer='adam')
lstm_model.fit(x_train_data,y_train_data,epochs=1,batch_size=1,verbo
se=2)X_test=[]
for i in range(60,model_data.shape[0]):
   X_test.append(model_data[i-60:i,0])
X_test=np.array(X_test)
X_test=np.reshape(X_test,(X_test.shape[0],X_test.shape[1],1))
```

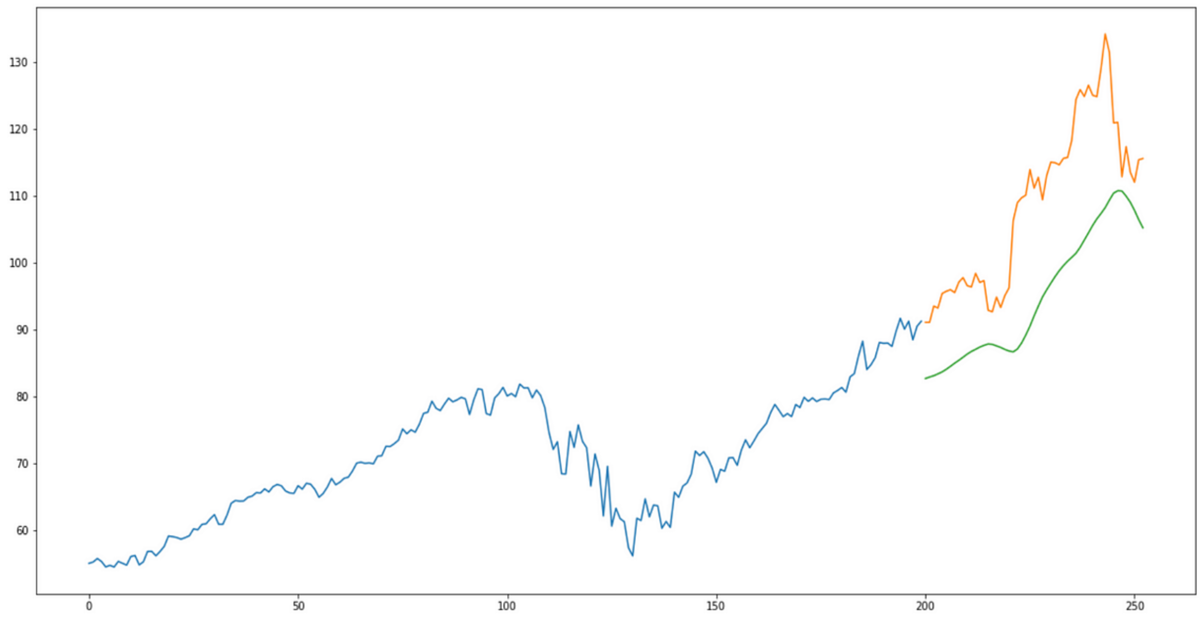
Prediction Function

```
predicted_stock_price=lstm_model.predict(X_test)
predicted_stock_price=scaler.inverse_transform(predicted_stock_price
)
```

Prediction Result

```
train_data=data[:200]
valid_data=data[200:]
valid_data['Predictions']=predicted_stock_price
plt.plot(train_data["Close"])
plt.plot(valid_data[['Close',"Predictions"]])
```

<u>OUTPUT:</u>



CONCLUSION:

- ❖ In conclusion, building a product demand prediction model with machine learning is a multifaceted process that requires a combination of feature engineering , data manipulation,domain knowledge ,modeling techniques like model training and model evaluation and also a careful evaluation.

- ❖ To building a product demand prediction model is a comprehensive and iterative process that blends data science techniques with domain knowledge and business understanding.

- ❖ Success in this endeavor depends on a combination of data quality, modeling expertise, and continuous refinement to ensure accurate and actionable predictions for decision-making.