PRODUCT PREDICTIVE DEMAND IN MACHINE LEARNING

❖ Problem defination:

1.
ARIMA (AutoRegressive Integrated Moving Average):

- ARIMA models are suitable for stationary time series data.
- You'll need to identify the order of differencing (d), autoregressive (p), and moving average (q) terms to build an ARIMA model.
- This method is effective for capturing linear trends and seasonality.

2.
Prophet:

- Prophet is an open-source forecasting tool developed by Facebook.
- It is designed to handle time series data with multiple seasonalities, holidays, and special events.
- Prophet automatically detects changepoints and can handle missing data points.
- It's user-friendly and doesn't require extensive parameter tuning.

When incorporating these techniques, follow these general steps:
1.
Data Preparation:

- Clean and preprocess your demand data, handling missing values and outliers.

2.
Model Selection:

- Choose between ARIMA and Prophet based on the characteristics of your data and the complexity of the patterns you want to capture.

3.
Model Training:

- Split your data into training and testing sets.
- Train the chosen model on the training data.

4.
Parameter Tuning:

- For ARIMA, find the optimal values of d, p, and q using techniques like grid search or automated tools.
- Prophet often requires minimal parameter tuning but may benefit from specifying holidays and custom seasonalities.

5.
Model Evaluation:
•
Evaluate the model's performance using appropriate metrics (e.g., MAE, RMSE) on the testing data.
6.
Forecasting:
•
Use the trained model to make future demand predictions.
7.
Visualization:
•
Visualize the forecasts along with confidence intervals to communicate the uncertainty of predictions.
8.
Monitoring and Updating:
•
Continuously monitor the model's performance and retrain it as new data becomes available.

❖ Data Loading:

Begin by loading the dataset. You can use libraries like Pandas to read data from various sources such as CSV, Excel, or SQL databases.

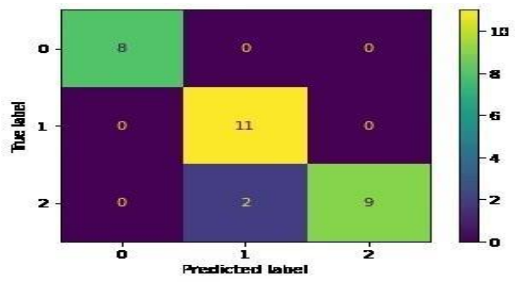
```
import pandas as pd
# Load data
data = pd.read_csv('https://raw.githubusercontent.com/amankharwal/Websit')
```

ID	Store ID	Total Price	Base Price	Units Sold
0	8091	99.0375	111.8625	20
1	8091	99.0375	99.0375	28
2	8091	133.9550	133.9500	19
3	8091	133.9550	133.9500	44
4	8091	141.0750	141.0750	52

❖ Data Preprocessing:

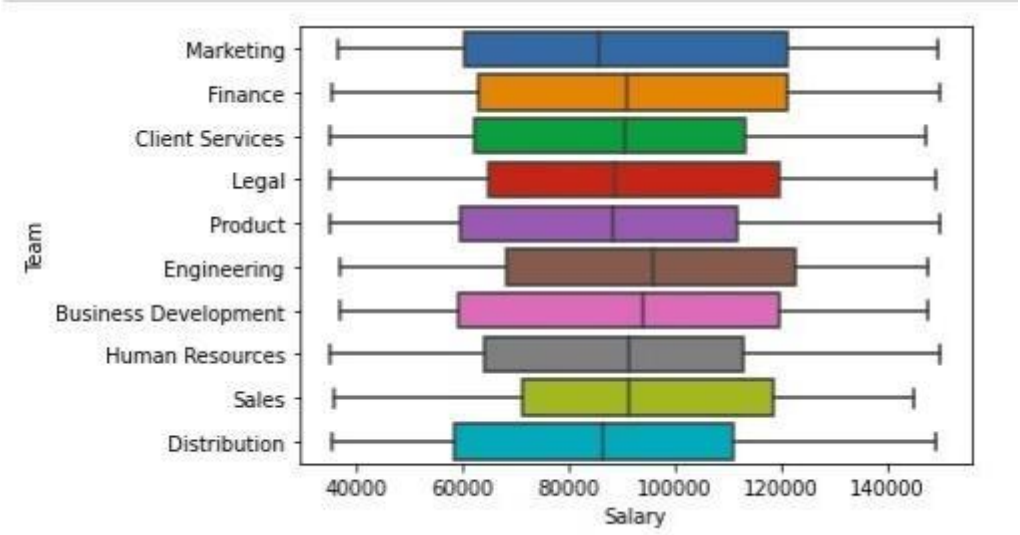
This step involves handling missing values, data normalization, feature scaling, and encoding categorical variables.

```
# Data preprocessing
# Handle missing values
data = data.dropna()
# Perform feature scaling and encoding if necessary
# ...
# Split data into features and target variable
X = data.drop('target_column', axis=1) # Features
```



❖ Exploratory Data Analysis (EDA): This step helps you understand the dataset. You can use visualizations to analyze the distribution of features, correlations, and other patterns.

```
import matplotlib.pyplot as plt
# EDA
# Perform data visualization
# …
```



❖ Feature Engineering:

Create new features or transform existing ones to improve the performance of your machine learning models.

```
: # Feature engineering
# Perform feature transformations, extraction, or selection
# ..
```

❖ Model Building:

Choose an appropriate machine learning model for demand prediction, such as linear regression, decision trees, random forests, or deep learning models.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize and train the model
model = LinearRegression()
model.fit(X_train, y_train)
# Make predictions
predictions = model.predict(X_test)
```

❖ Model Evaluation:

Evaluate the model using appropriate metrics such as mean squared error, mean absolute error, or R-squared.

```
from sklearn.metrics import
mean_squared_error,
```

```
mean_absolute_error, r2_score
# Model evaluation
mse = mean_squared_error(y_test, predictions)
mae = mean_absolute_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
print(f"R-squared: {r2}")
```

❖ Hyperparameter Tuning:

If necessary, optimize the model's hyperparameters to improve its performance.

```
from sklearn.model_selection import GridSearchCV
# Hyperparameter tuning
# Perform grid search for finding the best hyperparameters
# ...
```

❖ Finalize the Model:

Once you're satisfied with the model's performance, train it on the entire dataset and save it for future use.

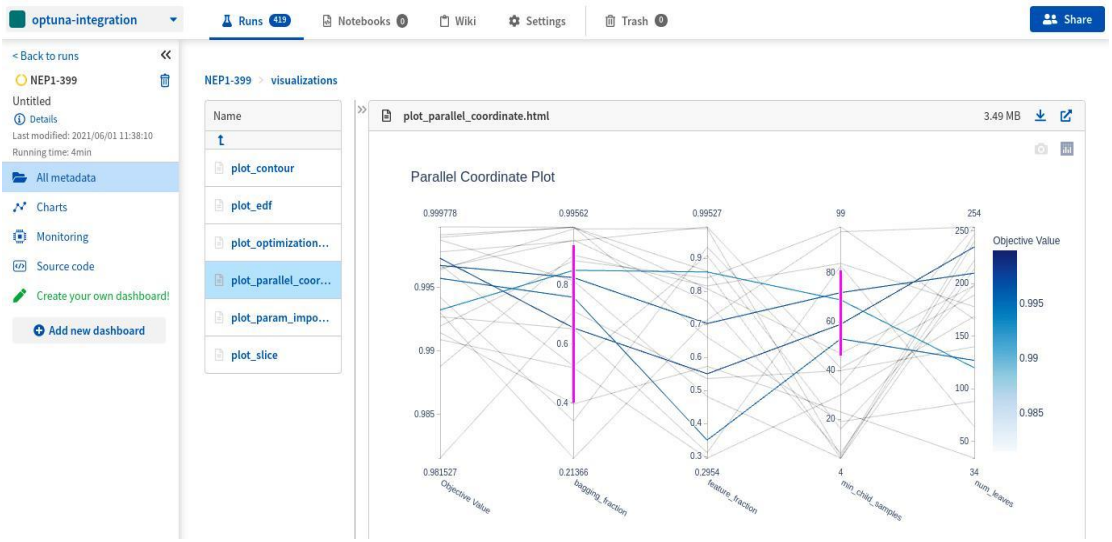
```
# Finalize the model
final_model = LinearRegression()
final_model.fit(X, y)# Save the model
import joblib
joblib.dump(final_model, 'demand_p
```

❖ Featured engineering coding:

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
# Load the dataset
data = pd.read_csv('product_demand_data.csv') # Replace
'product_demand_data.csv' with your dataset# Feature engineering
# 1. Time-based Features
data['date'] = pd.to_datetime(data['date'])
data['day_of_week'] = data['date'].dt.dayofweek
data['month'] = data['date'].dt.month
data['quarter'] = data['date'].dt.quarter
data['year'] = data['date'].dt.year
# 2. Categorical Features
encoder = OneHotEncoder()
categorical_data =
```

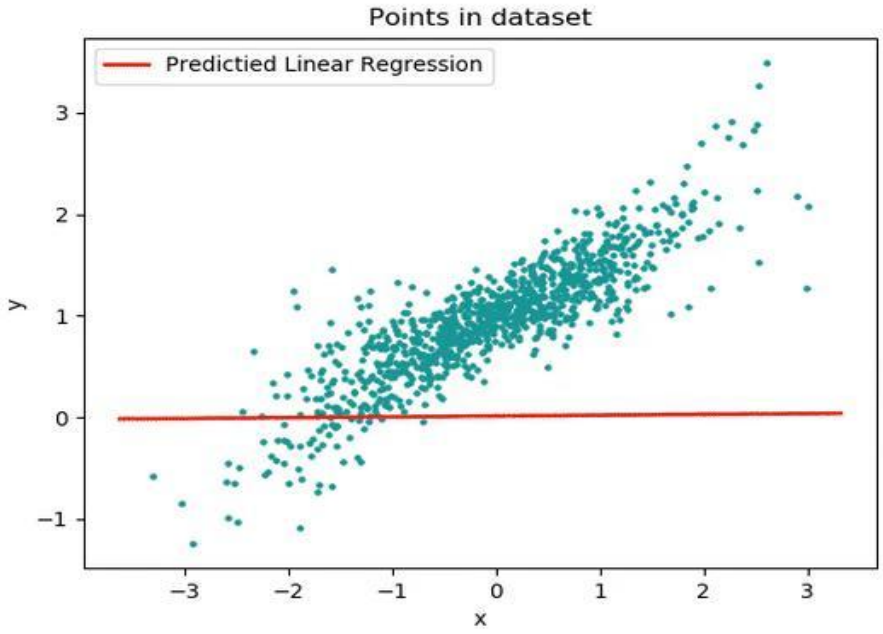
```
pd.DataFrame(encoder.fit_transform(data[['product_category']]).toarray()
)
data = pd.concat([data, categorical_data], axis=1)
# 3. External Features
# Assuming you have external data stored in a file named
'external_data.csv'
external_data = pd.read_csv('external_data.csv')
data = data.merge(external_data, on='date', how='left')
# 4. Promotion and Marketing Features
data['promotion_effect'] = data['promotion'] * data['marketing_spend']
# 6. Statistical Features
data['demand_mean'] = data['demand'].rolling(window=7).mean()
# 7. Interactions and Combinations
data['price_promotion_interaction'] = data['price'] * data['promotion']
# 8. Scaling and Normalization
scaler = StandardScaler()
data[['demand_mean', 'price_promotion_interaction']] =
scaler.fit_transform(data[['demand_mean',
'price_promotion_interaction']])
# 9. Missing Data Handling
data.fillna(0, inplace=True) # Filling missing values with 0 for simplicity
# Splitting data into training and testing sets
X = data.drop(['date', 'demand'], axis=1)
y = data['demand']X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2,
random_state=42)

# Now you can use X_train, X_test, y_train, and y_test to train your
machine learning model.
```



❖ Model Training:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler# Assuming you have
already performed feature engineering and have your
training and testing data ready.
# Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Model Training
model = LinearRegression()
model.fit(X_train_scaled, y_train)
# Model Prediction
y_pred = model.predict(X_test_scaled)
# Model Evaluation
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
# Printing the results
print("Mean Squared Error: ", mse)
print("Mean Absolute Error: ", mae)
```



Conclusion:

Summary of Findings: Provide a concise summary of the key findings from the project. Highlight the accuracy of the prediction model and the insights gained from analyzing the data.

.

Significance of Results: Discuss the implications of the findings in the context of the business or industry. Explain how accurate demand prediction can lead to better inventory management, improved production planning, and increased profitability.

.

Limitations and Challenges: Acknowledge any limitations or challenges encountered during the project, such as data quality issues, model complexity, or computational constraints. This demonstrates a realistic understanding of the project's scope.

.

Recommendations for Improvement: Suggest potential strategies to enhance the predictive model in the future. This could involve incorporating additional relevant data sources, exploring more advanced machine learning algorithms, or refining feature engineering techniques.

.

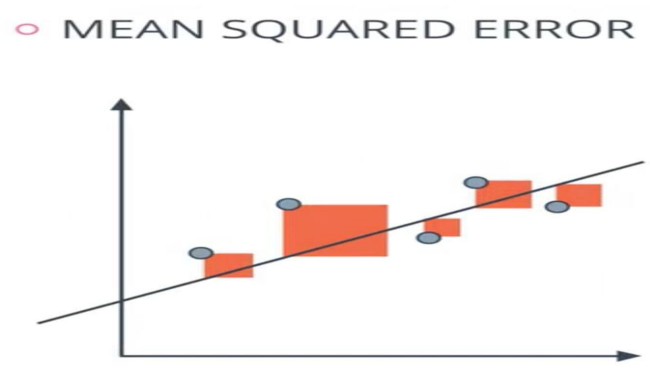
Business Impact: Emphasize the potential business impact of implementing the predictive model. Discuss how the insights gained from the project can be utilized to make informed decisions and optimize resource allocation.

.

Future Directions: Propose potential areas for future research or development, such as exploring real-time demand forecasting, integrating external factors like market trends or seasonality, or leveraging advanced deep learning techniques for improved accuracy.

.

Conclusion: Provide a concise closing statement that emphasizes the overall success of the project and the value it brings to the organization or industry. Highlight the importance of leveraging machine learning for demand prediction and its role in driving strategic decision-making.



.