PRODUCT PREDICTIVE DEMAND IN MACHINE LEARNING

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  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
```

Data Preprocessing:

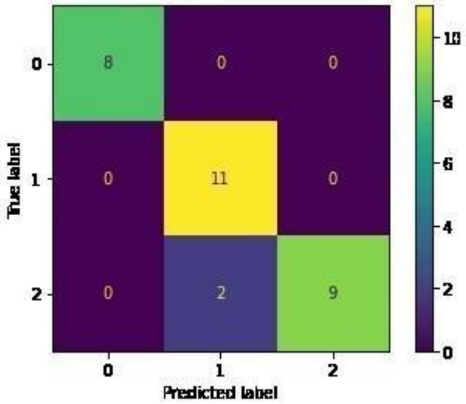
This step involves handling missing values, data normalization, feature scaling, and encoding categorical variables.

```
# 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
```

y = data['target_column'] #

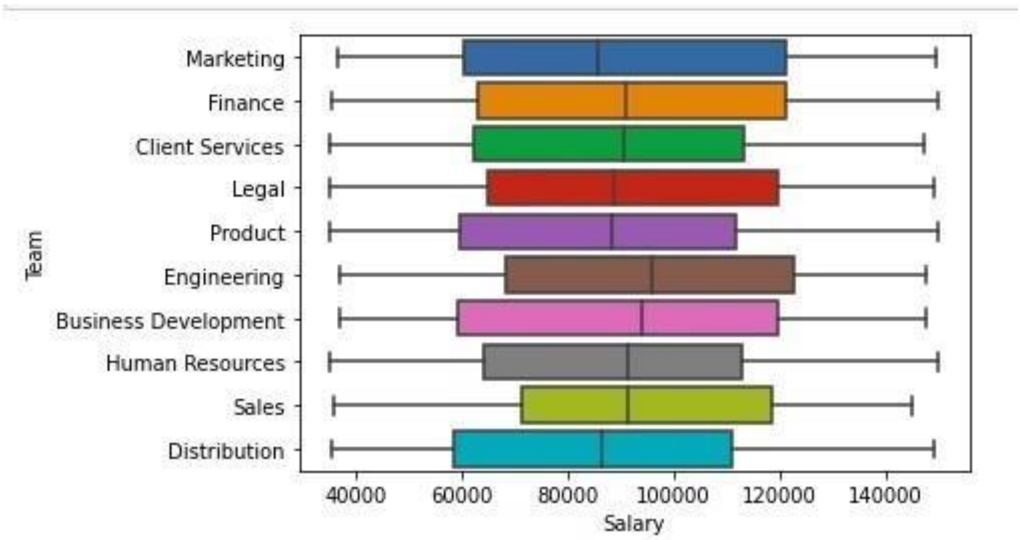


Targ et

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:

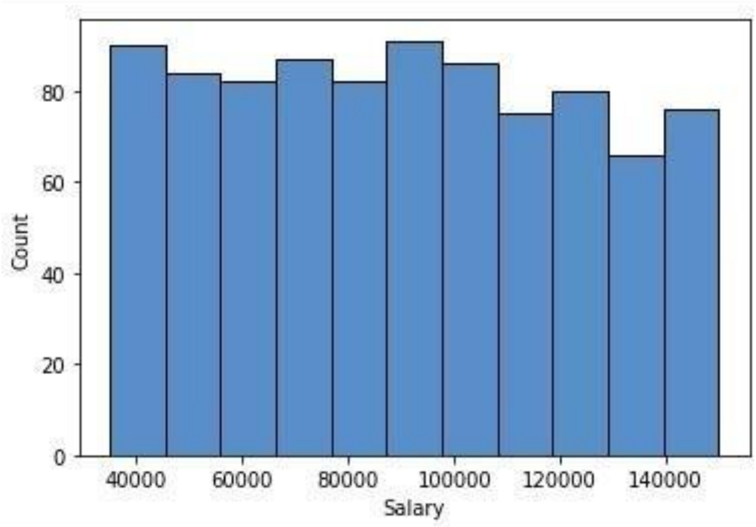
Choose an appropriate machine learning model for demand prediction, such as linear regression, decision trees, random forests, or deep learning models.

```
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)
```



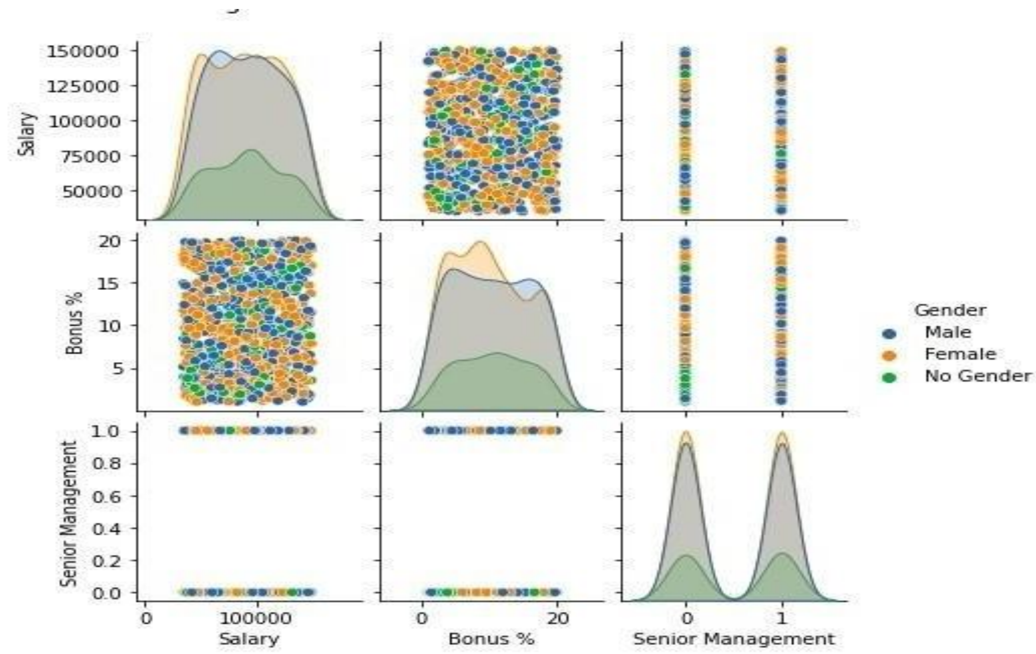
Model Evaluation:

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}")
```

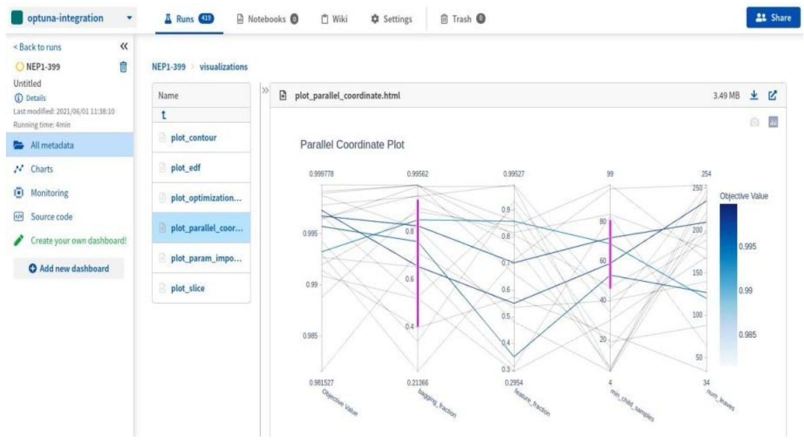


Hyperparameter Tuning:

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
```

Done by:
Divya sree
au720921244039
Jct college of engineering and techonology