PRODUCT SALES ANALYSIS USING MACHINE LEARNING

Phase 2 Submission Document

Project: Product Sales Analysis

Introduction:

- Sales analysis is the process of integrating, analyzing and understanding various data related to sales activities such as sales, customers, and transaction data.
- The product sales analysis provides insight into how well a product is matched to prospective customer needs. It can also indicate which products are the most successful, when and why certain products are most likely to be purchased, and which types of consumers are likely to buy a specific product or service.
- Product sales analysis has evolved to harness the power of machine learning, making it an even more potent tool for project success. Machine learning techniques enable businesses to extract deeper insights, predict future sales trends, and drive data-driven decision-making to a new level. When applied to a project, machine learning-based product sales analysis can deliver unparalleled advantages.

Content For Project Phase 2:

Consider incorporating machine learning algorithms to predict future sales trends or customer behaviors.

Data Source:

A good data source should be accurate,reliable and consistent.

Dataset Link:(Product Sales Data (kaggle.com))

Data Collection And Preprocessing:

- Importing the dataset: Obtain a comprehensive dataset containing relevant features.
- Data collection: It is the process of gathering data for use in business decision making, strategic planning, research and other purposes. It helps improve services, understand consumer needs, refine business strategies, grow and retain customers, and even sell the data as second party data to other businesses at a profit.
- Data pre-processing: It is the concept of changing raw data into a clean dataset. The dataset is preprocessed in order to check missing values, noisy data, and other inconsistencies before executing it to the algorithm.

Exploratory Data Analysis:

- Visualize and analyse the dataset to gain insights into the relationships between variables.
- Identify correlations and patterns that can inform feature selection and engineering.
- Present various data visualizations to gain insights into the dataset.
- Explore correlations between features and the target variable
- Discuss any significant findings from the EDA phase that inform feature selection.

Feature Engineering:

- Create new features or transform existing ones to capture valuable information.
- Utilize domain knowledge to engineer features that may cause impact.
- Explain the process of creating new features or transforming existing ones.
- Showcase domain specific feature engineering such as proximity scores or composite indicators.

- Emphasize the impact of engineered features on model performance.

Regression Techniques:

- Linear Regression: Linear regression is a powerful tool for understanding and predicting the behavior of a variable, however, it needs to meet a few conditions in order to be accurate and dependable solutions
- Lasso Regression: Employ L1 regularization to perform feature selection and simplify the model.
- A Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. SVMs are particularly effective for solving binary classification problems, but they can also be adapted for multi-class classification and regression tasks
- Random Forest Regression: Implement an ensemble technique to handle nonlinearity and capture complex relationships in the data.
- Decision Tree:A decision tree is a supervised machine learning algorithm that is used for both classification and regression tasks.

Forecasting:

- Estimate the future revenue by predicting how much of a product or service will sell in the next week, month, quarter or year.
- Detail analysis of past and present trends or events to predict future events.
- Project the measure of how a market will respond to a company's go to market efforts.
- A sales forecast must integrate a lot of data because the more important and qualified the data is the more accurate it will be.

Customer Segmentation:

- Create more specific sales and marketing strategies for customer groups.
- Discover insights that define specific segments of customers.
- Marketers and brands leverage this process to determine what campaigns, offers or products to leverage when communicating with specific segments.

- Divide a company's customers into groups based on common characteristics so company's can market to each other effectively and appropriately.
- Apply clustering techniques to segment customers based on their purchasing behaviour and tailor marketing techniques accordingly.

Anomaly Detection:

- Implement anomaly detection models to identify unusual sales patterns or fraudulent activities.
- Detect points on a given input time series where the behaviour isn't what was expected or weird.
- Identify rare events, items, observations which are suspicious because they differ significantly from standard behaviours or patterns.

Model Evaluation and Selection:

- Split the dataset into training and testing sets.
- Evaluate models using appropriate metrics (e.g., Mean Absolute Error, Mean Squared Error, R-squared) to assess their performance.
- Use cross-validation techniques to tune hyper parameters and ensure model stability.
- Look at more insightful statistics of its performance.
- Decide what actions to take in order to improve this model.
- Compare the results with traditional linear regression models to highlight improvements.
- Select the best-performing model for further analysis.

Model Interpretability:

- Explain how to interpret feature importance from Gradient Boosting and XG Boost models.
- Discuss the insights gained from feature importance analysis and their relevance to product sales analysis.
- Interpret feature importance from ensemble models like Random Forest and Gradient Boosting to understand the factors influencing product sales.

Business Insights:

- Translate the machine learning results into actionable insights for the business which can inform price, inventory management and marketing strategies.

Deployment Prediction:

- Deploy the chosen regression model to predict product sales analysis.
- Develop a user-friendly interface for users to input property features and receive price predictions.

Model 1- Linear Regression

Import the necessary libraries:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

Load your dataset into a Pandas DataFrame:

```
data=pd.read_csv("C:\\Users\\jacin\\Downloads\\products\\statsfinal.csv")
```

Split the data into features (X) and the target variable (y):

```
X = data[['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4']]
y = data['S-P1']
```

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)
```

Create a Linear Regression model and fit it to the training data

```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out: LinearRegression()

Make predictions on the test set:

```
y_pred = model.predict(X_test)
```

Evaluate the model's performance:

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R2): {r2}')
```

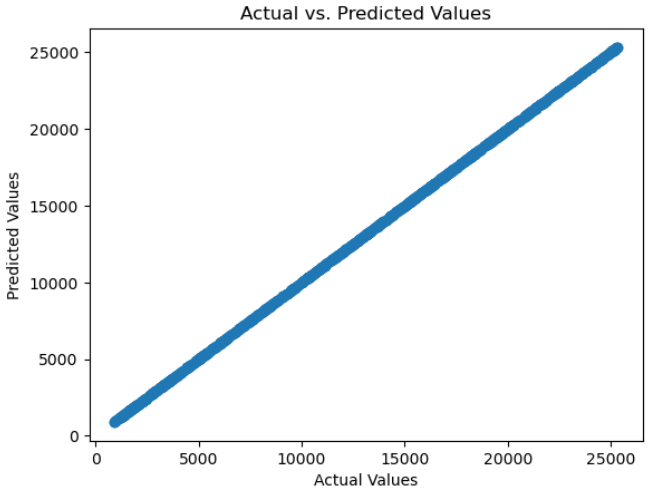
Out: Mean Squared Error (MSE): 1.139556492178734e-23

R-squared (R2): 1.0

Create a scatter plot to visualize the actual vs. predicted values:

```
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs. Predicted Values')
plt.show()
```

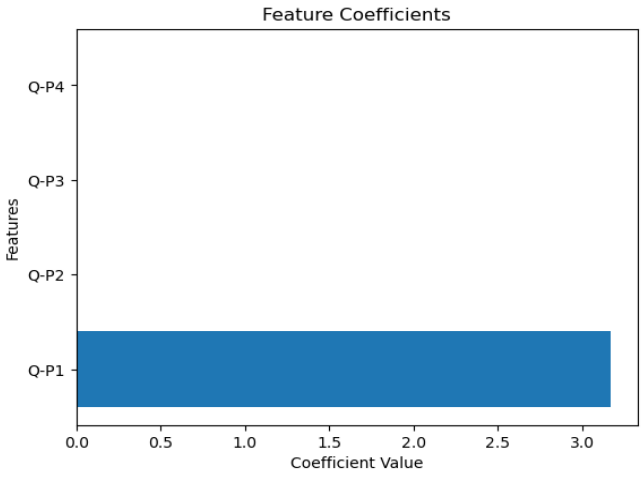
Out:



You can also visualize the coefficients of the model to understand the feature importance:

```
coefficients = model.coef_

feature_names = X.columns


plt.barh(feature_names, coefficients)

plt.xlabel('Coefficient Value')

plt.ylabel('Features')

plt.title('Feature Coefficients')

plt.show()
```

Out:



Model 2: Lasso Regression

Import the necessary libraries:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

Load your dataset into a Pandas DataFrame:

```
data=pd.read_csv("C:\\Users\\jacin\\Downloads\\products\\statsfinal.csv")
```

Split the data into features (X) and the target variable (y). In your case, you want to predict 'S-P1' (Total revenue from product 1) based on the other columns (Q-P1, Q-P2, Q-P3, Q-P4, S-P2, S-P3, S-P4).

```
X = data[['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4']]
y = data['S-P1']
```

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)
```

Create a Lasso Regression model and fit it to the training data:

```
alpha = 1.0
lasso_model = Lasso(alpha=alpha)
lasso_model.fit(X_train, y_train)
```

Out: Lasso()

Make predictions on the test set:

```
y_pred = lasso_model.predict(X_test)
```

Evaluate the model's performance:

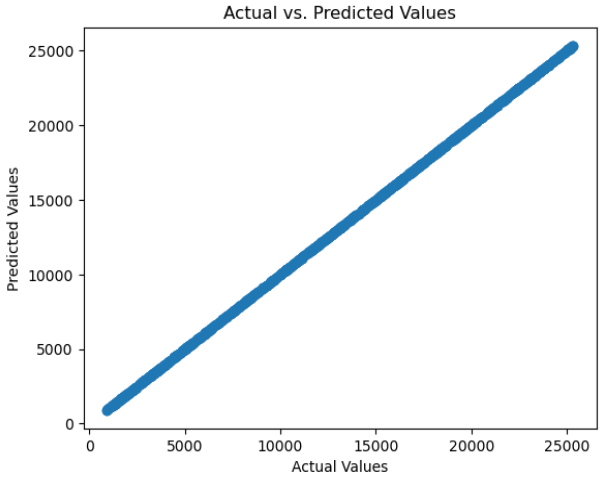
```
mse = mean_squared_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error (MSE): {mse}')

print(f'R-squared (R2): {r2}')
```

Out: Mean Squared Error (MSE): 2.016327108852503e-07
R-squared (R2): 0.999999999999996

Create a scatter plot to visualize the actual vs. predicted values:

```
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs. Predicted Values')
plt.show()
```

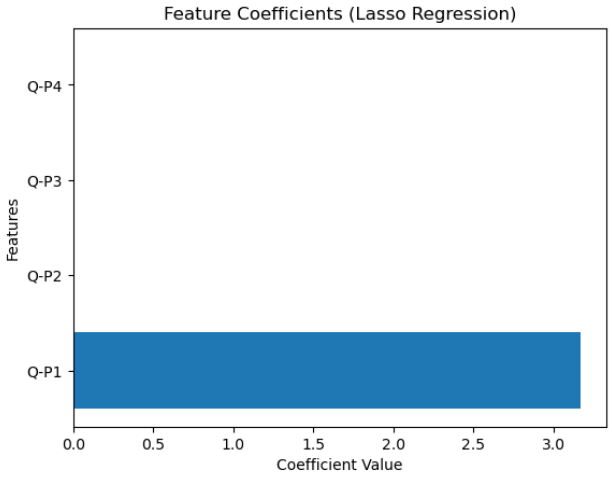
Out:



You can also visualize the coefficients of the model to understand the feature importance:

```
coefficients = lasso_model.coef_

feature_names = X.columns

plt.barh(feature_names, coefficients)

plt.xlabel('Coefficient Value')

plt.ylabel('Features')

plt.title('Feature Coefficients (Lasso Regression)')

plt.show()
```

Out:



Model 3: Support Vector machine

Import the necessary libraries:

```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.svm import SVR

from sklearn.metrics import mean_squared_error, r2_score
```

```
import matplotlib.pyplot as plt
```

Load your dataset into a Pandas DataFrame:

```
data=pd.read_csv("C:\\Users\\jacin\\Downloads\\products\\statsfinal.csv")
```

Split the data into features (X) and the target variable (y). In your case, you want to predict 'S-P1' (Total revenue from product 1) based on the other columns (Q-P1, Q-P2, Q-P3, Q-P4, S-P2, S-P3, S-P4

```
X = data[['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4', 'S-P2', 'S-P3', 'S-P4']]
y = data['S-P1']
```

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)
```

Create a Support Vector Machine Regression model and fit it to the training data:

```
svm_model = SVR(kernel='linear')
svm_model.fit(X_train, y_train)
```

Out: SVR(kernel='linear')

Make predictions on the test set:

```
y_pred = svm_model.predict(X_test)
```

Evaluate the model's performance:

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R2): {r2}')
```

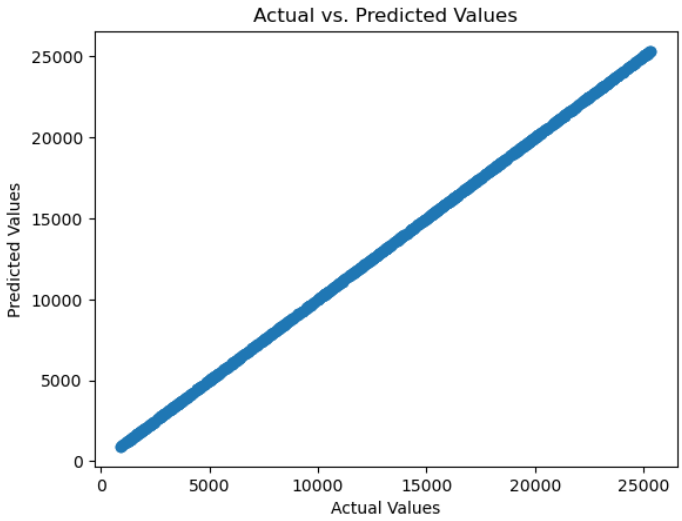
Out: Mean Squared Error (MSE): 0.0015957374427032214

R-squared (R2): 0.9999999999687661

Create a scatter plot to visualize the actual vs. predicted values:

```
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs. Predicted Values')
plt.show()
```

Out:



Model 4: Decision tree

Import the necessary libraries:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
```

```
from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import mean_squared_error, r2_score

import matplotlib.pyplot as plt
```

Load your dataset into a Pandas DataFrame:

```
data=pd.read_csv("C:\\Users\\jacin\\Downloads\\products\\statsfinal.csv")
```

Split the data into features (X) and the target variable (y):

```
X = data[['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4']]

y = data['S-P1']
```

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)
```

Create a Decision Tree Regressor model and fit it to the training data:

```
tree_model = DecisionTreeRegressor(random_state=42)

tree_model.fit(X_train, y_train)
```

Out:DecisionTreeRegressor(random_state=42)

Make predictions on the test set:

```
y_pred = tree_model.predict(X_test)
```

Evaluate the model's performance:

```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error (MSE): {mse}')
print(f'R-squared (R2): {r2}')
```

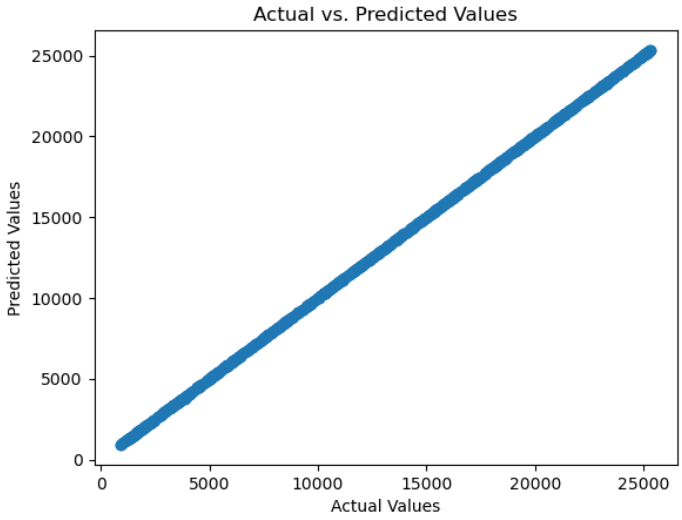
Out: Mean Squared Error (MSE): 107.02078499999966

R-squared (R2): 0.9999979052497234

Create a scatter plot to visualize the actual vs. predicted values:

```
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs. Predicted Values')
plt.show()
```

Out:



Model 5: Random Forest

Import the necessary libraries:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

Load your dataset into a Pandas DataFrame:

```
data=pd.read_csv("C:\\Users\\jacin\\Downloads\\products\\statsfinal.csv")
```

Split the data into features (X) and the target variable (y):

```
X = data[['Q-P1', 'Q-P2', 'Q-P3', 'Q-P4']]

y = data['S-P1']
```

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)
```

Create a Random Forest Regressor model and fit it to the training data:

```
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)  # You can adjust the number of trees (n_estimators)

rf_model.fit(X_train, y_train)
```

Out: RandomForestRegressor(random_state=42)

Make predictions on the test set:

```
y_pred = rf_model.predict(X_test)
```

Evaluate the model's performance:

```
mse = mean_squared_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error (MSE): {mse}')

print(f'R-squared (R2): {r2}')
```

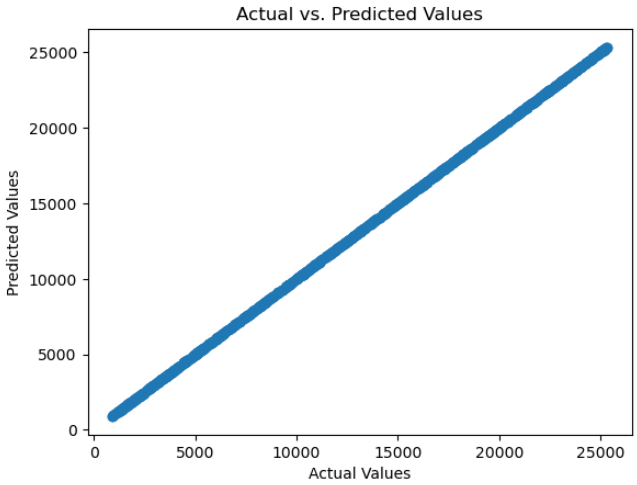
Out: Mean Squared Error (MSE): 29.94948159932641
R-squared (R2): 0.9999994137897151

Create a scatter plot to visualize the actual vs. predicted values:

```
plt.scatter(y_test, y_pred)

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')
```

```
plt.title('Actual vs. Predicted Values')
plt.show()
```

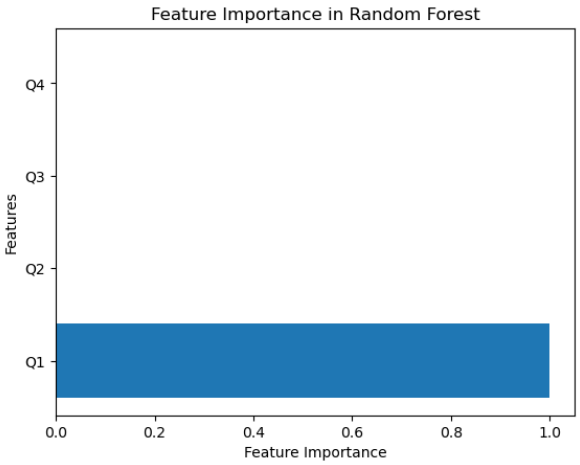
Out:



You can also investigate feature importance using the feature_importances_ attribute of the Random Forest model

```
feature_importance = rf_model.feature_importances_
feature_names = ['Q1', 'Q2', 'Q3', 'Q4']
plt.barh(feature_names, feature_importance)
plt.xlabel('Feature Importance')
plt.ylabel('Features')
plt.title('Feature Importance in Random Forest')
plt.show()
```

Out:



Conclusion

In phase 2 , we have summarized the key findings and insights from the advanced regression techniques . We will iterate the impact of these techniques on improving the accuracy and robustness of product sales analysis.

Machine learning models can provide valuable insights and predictive capabilities based on the dataset. The specific conclusions will depend on the results obtained from the models and the business goals that are aimed to achieve. It's essential to continuously monitor and update the models as new data becomes available to adapt to changing market conditions.