Sprint 6

Day 11 Task 1:

Sprints for AI
Use Case For Demand Forecasting Using POS Transaction Data

Team Demo

Team Members:

Tufail Irshad
Amjad Ali Malik
Zameer Ahmad Mir
Asif Ahmad Najar
Irfan Ahmad Mutoo

Sprint 6 Day 11 Task 1

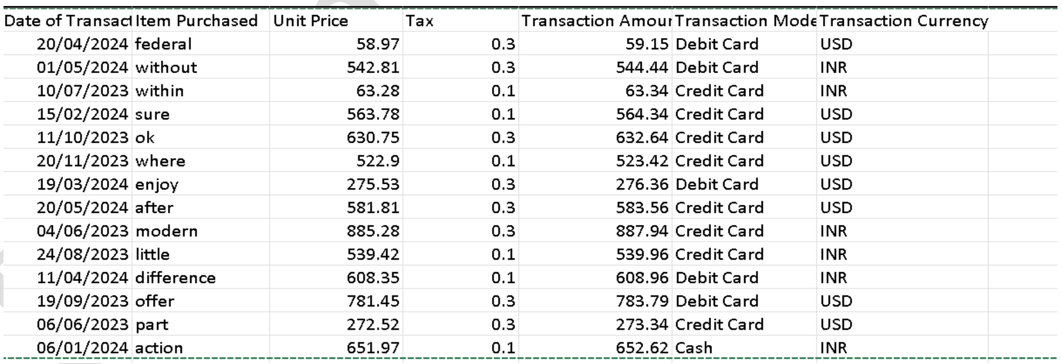
AI Comprehensive Project Closure Report

Project Overview:

The project aimed to generate synthetic Point of Sale (POS) transaction data using Faker library, preprocess the data, build and evaluate regression models to forecast transaction amounts, and provide future recommendations based on the analysis.

Project Achievements:

Synthetic Data Generation: Synthetic data consisting of 10,000 entries were generated with attributes such as transaction date, item purchased, unit price, tax, transaction amount, transaction mode, and transaction currency. The Faker library was utilized to create realistic-looking data that resembled actual POS transactions.



Date of Transact	Item Purchased	Unit Price	Tax	Transaction Amour	Transaction Mode	Transaction Currency
20/04/2024	federal	58.97	0.3	59.15	Debit Card	USD
01/05/2024	without	542.81	0.3	544.44	Debit Card	INR
10/07/2023	within	63.28	0.1	63.34	Credit Card	INR
15/02/2024	sure	563.78	0.1	564.34	Credit Card	USD
11/10/2023	ok	630.75	0.3	632.64	Credit Card	USD
20/11/2023	where	522.9	0.1	523.42	Credit Card	USD
19/03/2024	enjoy	275.53	0.3	276.36	Debit Card	USD
20/05/2024	after	581.81	0.3	583.56	Credit Card	USD
04/06/2023	modern	885.28	0.3	887.94	Credit Card	INR
24/08/2023	little	539.42	0.1	539.96	Credit Card	INR
11/04/2024	difference	608.35	0.1	608.96	Debit Card	INR
19/09/2023	offer	781.45	0.3	783.79	Debit Card	USD
06/06/2023	part	272.52	0.3	273.34	Credit Card	USD
06/01/2024	action	651.97	0.1	652.62	Cash	INR

Data Preprocessing:

Handling Null Values: Null values in the 'Transaction Amount' field were replaced with the mean transaction amount, ensuring that no data was lost during preprocessing.

Dynamo

- **Loading the Data**

The initial step involves loading your cleaned DataFrame (df_cleaned). This DataFrame is assumed to have no missing values or outliers.

```
import pandas as pd
dff=pd.read_csv('/content/pos_dataset.csv')
dff.head(5)
```

- **Encoding Categorical Columns**

Categorical columns are encoded using OneHotEncoder.

```
# Columns to encode and scale
categorical_columns = ['Item Purchased', 'Transaction Mode', 
'Transaction Currency']
# Encode categorical columns
def encode_categorical_columns(df, categorical_cols):
    # Initialize the OneHotEncoder
    ohe = OneHotEncoder(sparse=False, drop='first')
    encoded_df = 
pd.DataFrame(ohe.fit_transform(df[categorical_cols]), 
columns=ohe.get_feature_names_out(categorical_cols))

    # Concatenate the encoded columns back with the original 
DataFrame
    df = df.drop(categorical_cols, axis=1).reset_index(drop=True)
    df = pd.concat([df, encoded_df], axis=1)
    return df
```

- **Data cleaning**

Address missing values, outliers, and inconsistencies in the dataset through techniques such as outlier removal, and data transformation.

```
# Check for missing value
print(dff.isnull().sum())
Date of Transaction       0
Item Purchased            0
Quantity                 30
Unit Price                2
Tax                       0
Transaction Amount        0
Transaction Mode         25
Transaction Currency      2
Total Amount In INR       0
```

- **Date Format Correction:** The date format in the 'Date of Transaction' field was rectified to ensure consistency and compatibility with further analysis.
- **Removing Rows with Null Values:** Rows with null values in the 'Tax' column were dropped to maintain data integrity and accuracy.

```
# Find the most frequent value (mode) of the "Quantity" column
mode_quantity = dff["Quantity"].value_counts().idxmax()
# Fill missing values in the "Quantity" column with the mode
dff["Quantity"].fillna(mode_quantity, inplace=True)
```

-

- **Outlier Detection and Treatment:** Outliers in the 'Transaction Amount' were identified and treated using Z-Score. Entries with absolute Z-Scores greater than 3 were considered outliers and removed from the dataset. This process helped in improving the robustness of the regression models by reducing the influence of extreme values.
- **Detecting Outliers:**

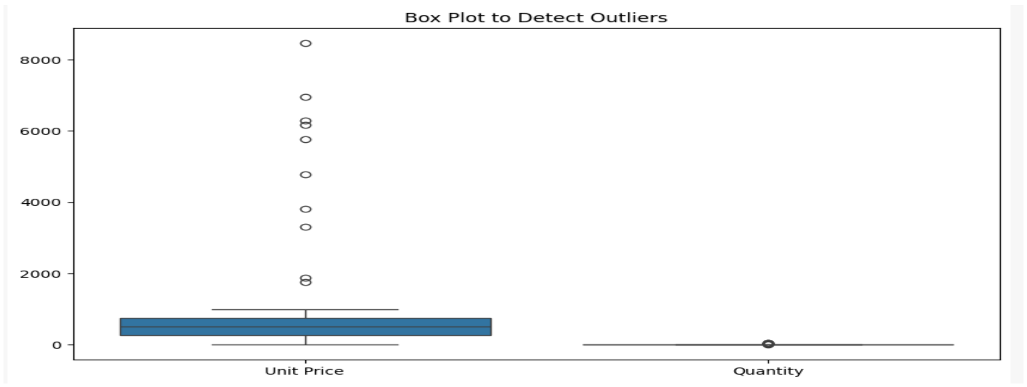
Box Plot: A box plot is used to visually identify outliers in the "Unit Price" and "Quantity" columns.

IQR Method: The Interquartile Range (IQR) method is used to detect outliers. Outliers are values that fall below IQR Method: The Interquartile Range (IQR) method is used to detect outliers. Outliers are values that fall below $Q1-1.5\times IQR$Q1−1.5×IQR or above $Q3+1.5\times IQR$Q3+1.5×IQR. The number of outliers detected is printedThe number of outliers detected is printed.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Detect outliers using a box plot
plt.figure(figsize=(10, 6))
sns.boxplot(data=dff[['Unit Price', 'Quantity']])
plt.title('Box Plot to Detect Outliers')
plt.show()
```

Team Dynamics



```
IQR Method
# Detect outliers using the IQR method
Q1 = dff[['Unit Price', 'Quantity']].quantile(0.25)
Q3 = dff[['Unit Price', 'Quantity']].quantile(0.75)
IQR = Q3 - Q1

outliers = ((dff[['Unit Price', 'Quantity']] < (Q1 - 1.5 *
IQR)) | (dff[['Unit Price', 'Quantity']] > (Q3 + 1.5 *
IQR))).any(axis=1)
print(f"Number of outliers detected: {outliers.sum()}")
```

```
Number of outliers detected: 20
```

- **Handling Outliers Using IQR**

```
# Function to remove outliers using IQR
def remove_outliers(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[col] >= lower_bound) & (df[col] <=
upper_bound)]
    return df
```

Model Building and Evaluation:

- **Regression Models:** Five regression models were trained and evaluated: Linear Regression, Lasso Regression, Decision Tree Regressor, Random Forest Regressor, and Support Vector Regression (SVR).
- **Evaluation Metrics:** The models were evaluated based on Mean Absolute Error (MAE) and R-squared (R^2) scores on both training and testing data. These metrics provided insights into the accuracy and performance of each model in predicting transaction amounts.
- **Splitting the Data**

Split the data into training and test sets

```
from sklearn.model_selection import train_test_split

df_preprocessed = pd.read_csv('preprocessed_data.csv')

# Split the data into features (X) and target (y)
X = df_preprocessed.drop(['Transaction Amount','Date of
Transaction'], axis=1)
y = df_preprocessed['Transaction Amount']
```

- **Training Multiple Models**

Train and evaluate multiple models to identify the best performer.

```
import pandas as pd
from sklearn.model_selection import train_test_split,
GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest": RandomForestRegressor(),
    "Gradient Boosting": GradientBoostingRegressor(),
    "Support Vector Machine": SVR(),
```

Team ...namo

```
}
results = {}
```

- **Error Calculation**

Calculate the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) to assess model accuracy.

```
for model_name, metrics in results.items():
    print(f"Model: {model_name}")
    print(f"  MSE: {metrics['MSE']}")
    print(f"  RMSE: {metrics['RMSE']}")
    print(f"  R-squared: {metrics['R-squared']}")
    print()

# Select the best model based on RMSE (or any other preferred
metric)
best_model_name = min(results, key=lambda x: results[x]['RMSE'])
best_model = models[best_model_name]

print(f"Best Model: {best_model_name}")
```

```
Model: Linear Regression
MSE: 5.494355140883775e-09
RMSE: 7.41239174685457e-05
```

Train Score of Different Models:

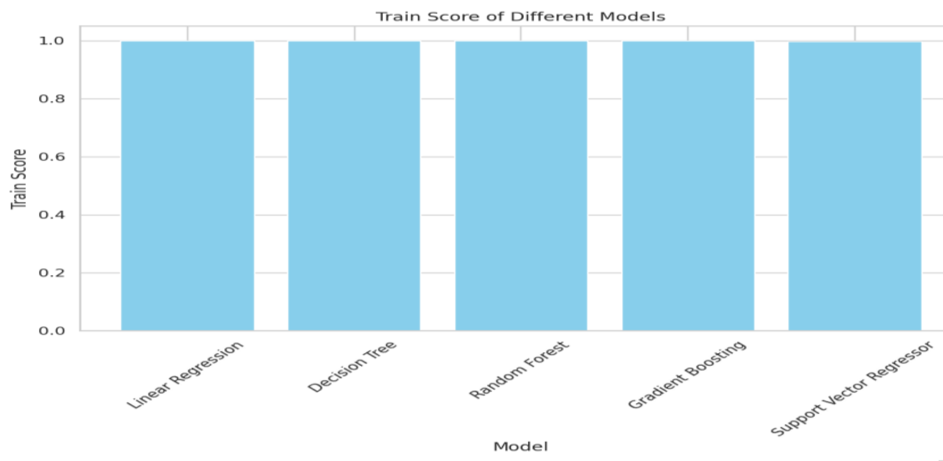
This plot shows the train score (also known as training accuracy) of different machine learning models.

The train score indicates how well each model fits the training data.

Higher values indicate better performance on the training data.

This plot helps in comparing the performance of models in capturing the patterns present in the training data.

Team D



- **Test Score of Different Models:**

  This plot displays the test score (also known as test accuracy) of different machine learning models.

  The test score represents how well each model generalizes to unseen data.

  Higher values suggest better performance on unseen data.

  It allows for comparison of models in terms of their ability to make accurate predictions on new data.

  - **R-squared of Different Models:**

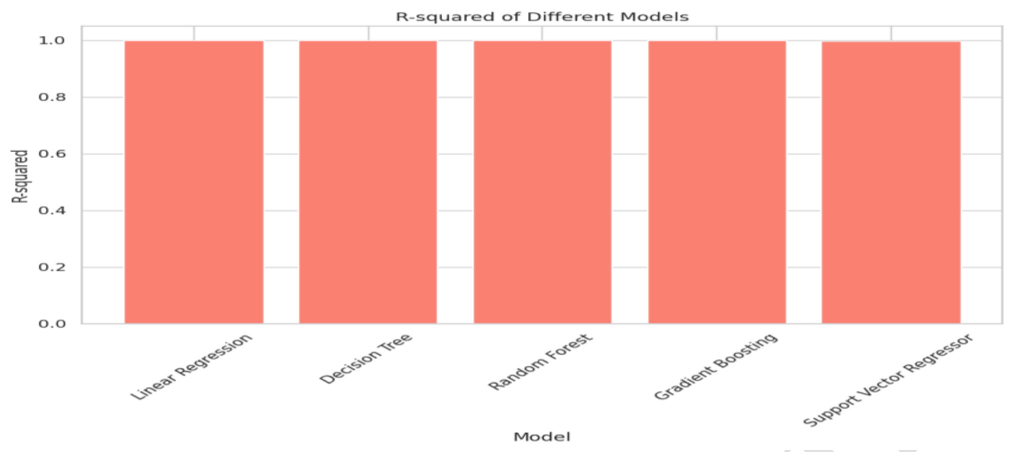
    R-squared (R^2) measures the proportion of the variance in the dependent variable that is predictable from the independent variables.

    This plot illustrates the R-squared values for each model, indicating the goodness of fit of the model to the data.

    Higher R-squared values indicate a better fit of the model to the data.

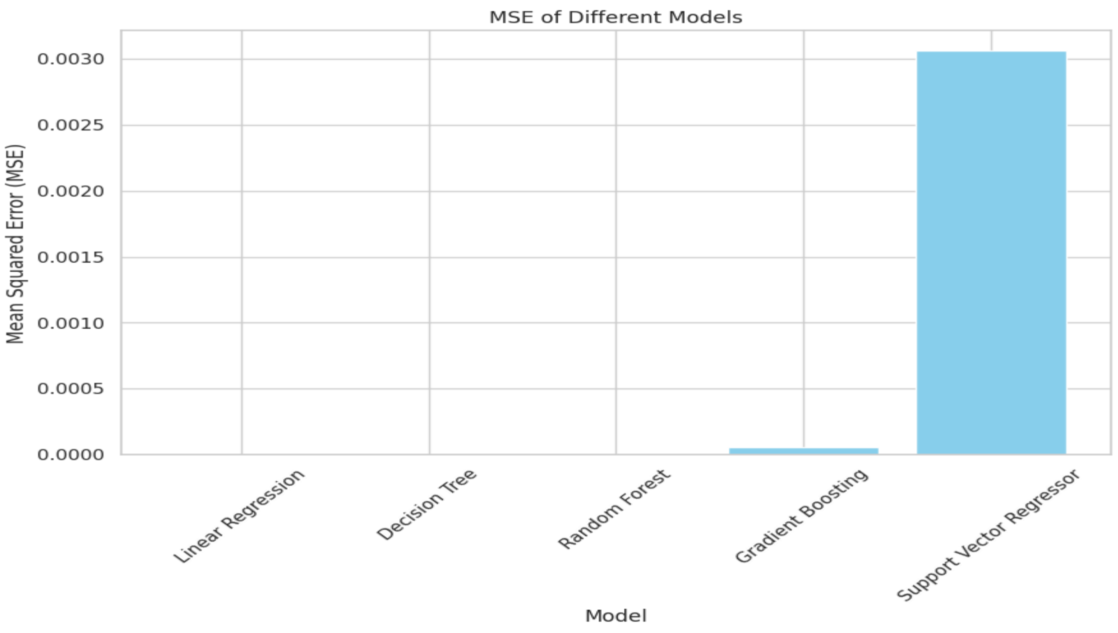
Team Dynamo

It provides insight into how well the independent variables explain the variability in the dependent variable.



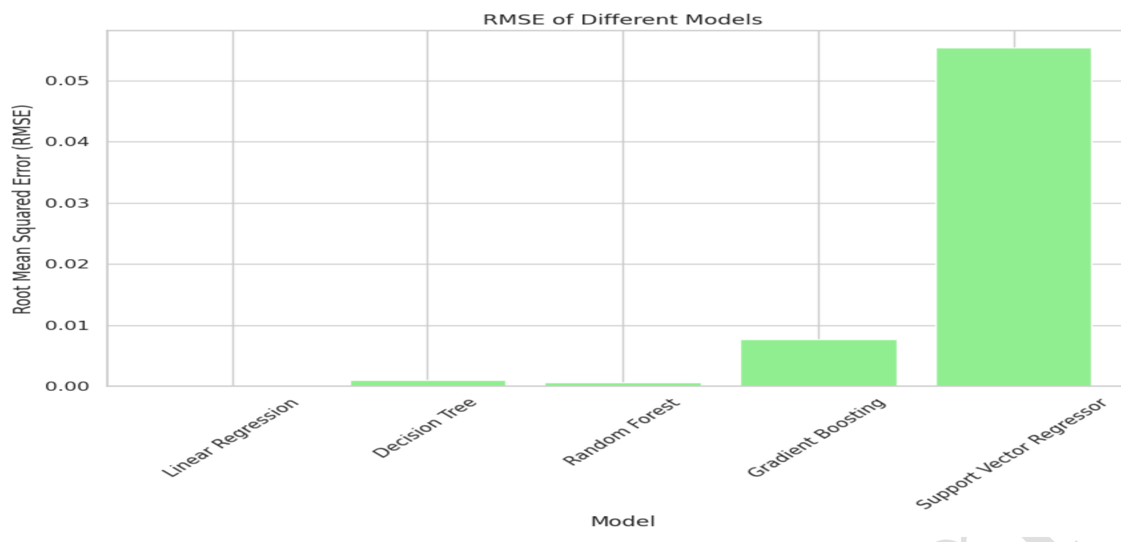
- **Mean Squared Error (MSE) of Different Models:**
    - Mean Squared Error (MSE) measures the average of the squares of the errors (residuals) between predicted and actual values.
    - This plot shows the MSE values for each model, representing the average squared difference between predicted and actual values.
    - Lower MSE values indicate better predictive performance of the model.
    - It helps in assessing the accuracy of predictions made by different models.

Team Dynamics



- **Root Mean Squared Error (RMSE) of Different Models:**
  - Root Mean Squared Error (RMSE) is the square root of the average of the squared differences between predicted and actual values.
  - This plot displays the RMSE values for each model, providing a measure of the average magnitude of errors in predictions.
  - Lower RMSE values indicate better performance in terms of prediction accuracy.
  - It offers insights into the overall performance of the models in making predictions.
  - These plots collectively offer a comprehensive evaluation of the performance of different machine learning models, facilitating the selection of the best model for a given task.

Team Dyna



Assessing Model Performance on Synthetic Data

Evaluate the model's performance using the synthetic data as a proxy for actual data.

```
# Calculate evaluation metrics for the new data

mse_new_data = mean_squared_error(df_preprocessed['Transaction
Amount'][:100], new_data['Predicted Transaction Amount'])
rmse_new_data = np.sqrt(mse_new_data)
r2_new_data = r2_score(df_preprocessed['Transaction
Amount'][:100], new_data['Predicted Transaction Amount'])
# Display the evaluation metrics
print("Evaluation metrics for new data:")
print(f"  MSE: {mse_new_data}")
print(f"  RMSE: {rmse_new_data}")
print(f"  R-squared: {r2_new_data}")
```

```
Evaluation metrics for new data:
  MSE: 2.114131487869145
  RMSE: 1.4540053259424963
  R-squared: -1.330250012350632
```

Forecasting Future Transactions:

Using the selected best model, the script generates forecasted transaction amounts for future dates. It does this by providing the model with features corresponding to those future dates (such as day of the week and month). The model then predicts the transaction amounts for those dates.

```
# Forecast demand for new data using the best model
new_dates = pd.date_range(start='2024-04-17', periods=100, freq='D')
new_features = pd.DataFrame({
    'DayOfWeek': new_dates.dayofweek,
    'Month': new_dates.month
})
new_predictions = best_model.predict(new_features)

forecast_df = pd.DataFrame({
    'Date': new_dates,
    'Predicted Transaction Amount': new_predictions
})
# Write the forecasted results to a new CSV file
forecast_df.to_csv('forecasted_results.csv', index=False)
print("Forecasted results saved to 'forecasted_results.csv'")
```

Date	Predicted Transaction Amount
17/04/2024	494.3628997
18/04/2024	493.6556228
19/04/2024	492.9483459
20/04/2024	492.241069
21/04/2024	491.5337921
22/04/2024	495.7774534

Team

Actual Values and Predicted Values

Actual values in machine learning represent the true outcomes or observations of the target variable in a dataset. These values are known and serve as the ground truth against which the model's predictions are compared. For example, in a regression task predicting house prices, the actual values would be the real sale prices of the houses in the dataset.

Actual Amount	Predicted Amount
479.92	495.7775
625.62	499.4513
654.3	499.5945
430.15	499.0275

Future Recommendations:

- **Data Quality Improvement**: Collecting more diverse and real-world data can enhance model performance and generalizability. Ensuring data cleanliness and accuracy is crucial for reliable predictions.
- **Feature Engineering:** Exploring additional features like customer demographics, seasonal trends, or promotional activities could improve model accuracy. Incorporating more relevant features can capture underlying patterns in transaction data more effectively.
- **Model Tuning:** Fine-tuning hyperparameters of the regression models might further enhance their performance. Experimenting with different parameter combinations can optimize model fitting and improve prediction accuracy.
- **Ensemble Methods**: Experimenting with ensemble methods like Gradient Boosting or Stacking could potentially improve prediction accuracy by combining the strengths of multiple models. Ensemble techniques often lead to more robust and reliable predictions compared to individual models.

Team Dynamo

- **Integration with Real Systems:** Integrating the developed forecasting model with real POS systems can provide real-time insights for business decision-making. Connecting the model to live data streams enables continuous monitoring and adaptation to changing transaction patterns.
- **Continuous Monitoring:** Regular monitoring of model performance and updating it with new data can ensure its relevance and accuracy over time. Implementing a feedback loop for model improvement based on real-world performance is essential for long-term success.

Conclusion:

The project successfully achieved its objectives by generating synthetic POS transaction data, preprocessing it to handle null values and outliers, building regression models for transaction amount prediction, and providing actionable recommendations for future improvements. By implementing the suggested recommendations, the project outcomes can be further enhanced, leading to better decision-making and business outcomes in the realm of POS

Team Dynamo

Gantt Chart

Sprints	Days Worked on Each Sprint	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Total Man-Days
Synthetic Data Generation	2	X	X													2
Data Cleaning	2			X	X											2
Feature Engineering	1					X										1
Model Training							X	X								2
Model Application and Iteration									X							1
Minimization of Bias and Overfitting										X						1
Test Models on Unseen Data or Validation Set											X	X				2
Final Model Selection and Conclusion													X			1
Documentation														X	X	2
Total Days		X	X	X	X	X	X	X	X	X	X	X	X	X	X	14

Te