Sprint 3

Day 5 Task 1:

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

Team Members:

Tufail Irshad

Amjad Ali Malik

Zameer Ahmad Mir

Asif Ahmad Najar

Irfan Ahmad Mutoo

Sprint 3 Day 5 Task 1

Report of Machine Learning Model And Training Code

Overview:

The code performs a machine learning task for regression using Python and scikit-learn. It starts by preparing the data, splitting it into training and testing sets, and standardizing the features. Then, it defines several regression models and trains each model using the training data. After training, it evaluates the performance of each model using Mean Squared Error (MSE) on the testing data.

Detailed Explanation:

Data Preparation:

- The code first ensures the correctness of column names in the DataFrame df_cleaned.
- It converts the 'Date of Transaction' column to a datetime object and then to an integer representation (Unix timestamp), likely for usability in machine learning models.
- It splits the data into features (X) and the target variable (y) for regression.

```
mport pandas as pd

# Assuming 'df_cleaned' is your DataFrame containing the dataset

# Check the column names to ensure correctness
print(df_cleaned.columns)

# Convert 'Date of Transaction\tItem' column to datetime object
df_cleaned['Date of Transaction'] = pd.to_datetime(df_cleaned['Date of
Transaction']).astype(int)

# Convert datetime object to integer representation (Unix timestamp)
df_cleaned['Date of Transaction'] = df_cleaned['Date of
Transaction'].astype(int)
# Splitting data into features and target
```

```
X = df_cleaned[['Unit Price', 'Tax', 'Total Amount In INR',
'Quantity']]
y = df_cleaned['Transaction Amount']
```

```
Index(['Date of Transaction', 'Item Purchased', 'Quantity', 'Unit Price',
       'Tax', 'Transaction Amount', 'Transaction Mode', 'Transaction Currency',
       'Total Amount In INR'],
      dtype='object')
<ipython-input-69-aa5c8c014e2d>:9: UserWarning: Parsing dates in %d/%m/%Y format when
dayfirst=False (the default) was specified. Pass `dayfirst=True` or specify a format to silence this warning.
  df_cleaned['Date of Transaction'] = pd.to_datetime(df_cleaned['Date of Transaction']).astype(int)
```

Data Splitting:

- The dataset is split into training and testing sets using an 80-20 split, with 80% of the data used for training and 20% for testing.

```
# Splitting 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)
```

Feature Scaling:

- Features are standardized using StandardScaler to ensure all features have a mean of 0 and a standard deviation of 1, making them comparable and suitable for many machine learning algorithms.

```
# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Model Definition:

- Several regression models are imported from scikit-learn, including Linear Regression, Decision Tree Regressor, Random Forest Regressor, Gradient Boosting Regressor, and Support Vector Machine Regressor.

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,
GradientBoostingRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error

# Define models
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest": RandomForestRegressor(),
    "Gradient Boosting": GradientBoostingRegressor(),
    "Support Vector Machine": SVR(),
}
```

Model Training and Evaluation:

- Each regression model is trained using the training data.
- Predictions are made on the testing data.
- Mean Squared Error (MSE) is calculated to evaluate the performance of each model.
- Model performance (MSE) is printed for each model.

```
# Train and evaluate each model
for name, model in models.items():
    # Train the model
    model.fit(X_train_preprocessed, y_train)

    # Predict on the testing set
    y_pred = model.predict(X_test_preprocessed)
```

```
    # Calculate Mean Squared Error
    mse = mean_squared_error(y_test, y_pred)

    # Print model performance
    print(f"{name}: Mean Squared Error = {mse}")
```

```
Linear Regression: Mean Squared Error = 0.009241841502682 42
Decision Tree: Mean Squared Error = 1.6252238977955948
Random Forest: Mean Squared Error = 0.5830579433216159
Gradient Boosting: Mean Squared Error = 98.70074166614376
Support Vector Machine: Mean Squared Error = 682220.2152943088
```

Overall, the code demonstrates a typical workflow for building, training, and evaluating machine learning regression models using Python and scikit-learn. It ensures that the data is properly prepared, models are trained, and their performance is assessed to select the best model for the given task.

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																
Test Models on Unseen Data or Validation Set																
Minimization of Bias and Overfitting																
Final Model Selection and Conclusion																
Documentation																

Total Days		X	X	X	X	X	X	X							14