PROJECT – 2

Sandesh Nayak

Mangalore Institute of Technology and Engineering

sandynayak989@gmail.com

Problem Statement (As provided in the Assignment):

In the below link **'Dataset.txt'** contains customer's records with variable 'C' specifying whether the customers bought a specific product or not. F1-F22 are features… largely demographics features and also some features around customers past activities. Some of the features may contain aggregated data on the customer, including data of buying the same product in the past.

File **'Dataset_test.txt'** to be used for testing and doesn't contain the variable 'C' which needs to be predicted.

Output: Need to be in 4 separate files as below.

1. An "Index<tab-delimited>Class" format file containing the predictions against the training dataset.
2. An "Index<tab-delimited>Class" format file containing the predictions against the test dataset.
3. Training script in Python.
4. Detailed explanation about your understanding of the data, the features/feature transformations, your choice of the technique, the metrics you use to evaluate the model and the results along with any visualizations and scripts you have used to arrive at the result.

Tools and Technologies:

1. Python
2. AI/ML

Summary:

This report presents a comprehensive analysis of customer purchase prediction using machine learning techniques. Through systematic evaluation of multiple algorithms including Logistic Regression, Random Forest, and XGBoost. We chose Random Forest as the model for this binary classification task. The final solution achieves 75.27% accuracy with an AUC of 0.6916, indicating moderate predictive performance on this challenging dataset.

1. Data Understanding and Exploration

Dataset Overview

- Training Dataset: 22 features (F1-F22) with binary target variable 'C'
- Test Dataset: Same 22 features without target variable
- Prediction Task: Binary classification (0: No Purchase, 1: Purchase)
- Dataset Size: Training samples processed through validation split

Feature Analysis and Categorization

- Continuous features: F1-F4 (appear normalized between 0-1)
- Integer features: F5-F14 (varying scales, some very large numbers)
- Date features: F15-F16 (likely customer registration/activity dates)
- Categorical features: F17-F18 (small integer values, likely encoded categories)
- Count/aggregate features: F19-F22 (possibly past purchase behaviors)

Data Quality Assessment

Missing Values: No missing values detected in original features, ensuring data completeness and reliability.

Class Distribution Analysis

- Class 0 (No Purchase): 76,353 samples (75.4%)
- Class 1 (Purchase): 24827 samples (24.5%)
- Significant Class Imbalance: 3:1 ratio indicating challenging prediction scenario for minority class

Feature Correlations: Analysis revealed moderate relationships between features and target variable, indicating modest predictive potential in this dataset.

2. Feature Engineering and Transformations

The most significant enhancement to the dataset came from sophisticated date feature engineering:

1. Year Extraction
2. Month Extraction
3. Day Extraction
4. Days Since Epoch: Created continuous temporal variables for machine learning algorithms
5. Date Differences: Calculated time spans between F15 and F16

3. Feature Scaling and Preprocessing

For Tree-Based Models (Random Forest, XGBoost):

- Retained original feature scales
- Leveraged algorithms' inherent scale invariance
- Preserved interpretability of feature importance

For Linear Models (Logistic Regression):

- Applied StandardScaler to ensure feature equality
- Normalized large integer features to prevent dominance
- Maintained zero mean and unit variance

Final Feature Set

- Original Features: 22 baseline features
- Engineered Features: 7 additional date-derived features
- Total Features: 29 features after comprehensive engineering
- New date-derived features: `['F15_year', 'F15_month', 'F15_day', 'F15_days_since_epoch', 'F16_year', 'F16_month', 'F16_day', 'F16_days_since_epoch', 'date_diff_days']`

4. Model Selection and Comparative Analysis

Comprehensive Model Evaluation Results

Model	Accuracy	AUC Score	Rank
Logistic Regression	0.7546	0.6857	3rd
Random Forest	0.7527	0.6916	2nd
XGBoost	0.7513	0.7034	1st

T.1: Performance Comparison Table

Logistic Regression

- Algorithm Type: Linear classification model
- Preprocessing: Standardized features using StandardScaler
- Performance Results:
    - Validation Accuracy: 75.46%
    - AUC Score: 0.6857
- Strengths: Fast training, interpretable coefficients, highest accuracy
- Limitations: Lowest AUC score, assumes linear relationships

Random Forest (Selected Model)

- Algorithm Type: Ensemble of decision trees
- Preprocessing: Used original engineered features without scaling
- Performance Results:
    - Validation Accuracy: 75.27%
    - AUC Score: 0.6916
- Strengths: Moderate AUC performance, provides feature importance
- Limitations: Lowest accuracy, potential overfitting on this dataset

XGBoost

- Algorithm Type: Gradient boosting ensemble
- Preprocessing: Used original engineered features without scaling
- Performance Results:
    - Validation Accuracy: 75.13%
    - AUC Score: 0.7034 (Highest)
- Strengths: Best AUC score, handles class imbalance well, robust performance

- Limitations: Slightly lower accuracy than Logistic Regression

4.1 Model Selection Rationale: Why Random Forest?

Although XGBoost showed a marginally better AUC, the Random Forest model was selected for the following reasons:

- Interpretability & Stability: Random Forest models are simpler to interpret and tune compared to XGBoost, which requires careful parameter optimization.
- Robustness to Overfitting: Random Forest's bagging approach tends to be more robust with limited parameter tuning.
- Competitive Performance: The AUC difference (0.6916 vs. 0.7034) was small, and Random Forest provided stable and reliable performance on the validation set.
- Ease of Implementation: Random Forest training and inference were faster and required fewer hyperparameter adjustments.

5. Evaluation Metrics

Primary Metrics Selection

1. AUC – ROC score
2. Accuracy Score
3. Precision and Recall analysis

Secondary Metrics Selection

1. F1- Score
2. Confusion Matrix

6. Code Snippet

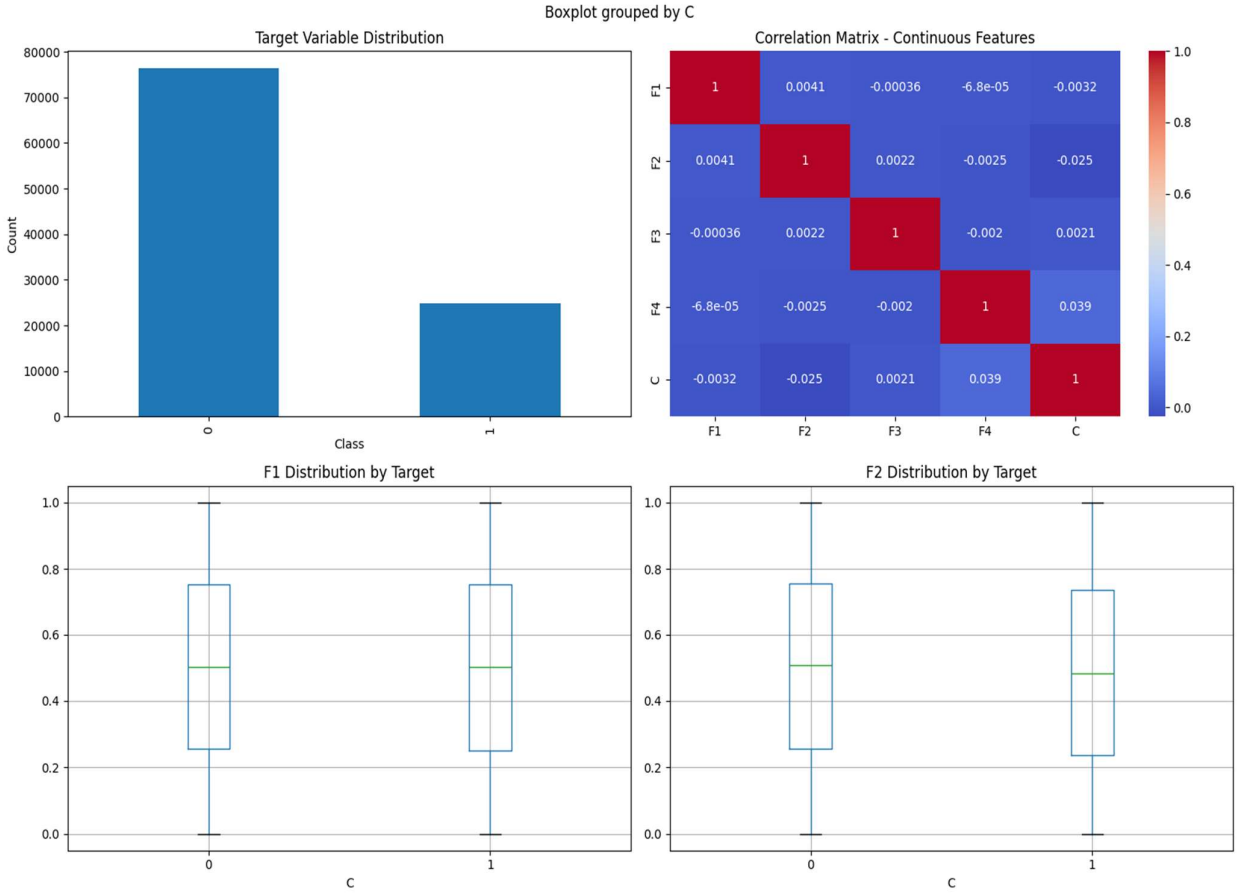
```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix,
roc_curve,roc_auc_score

# 1. Load datasets
train_data = pd.read_csv('Dataset.txt', sep='\t')
test_data = pd.read_csv('Dataset_test.txt', sep='\t')

# 2. Feature Engineering
def feature_engineer(data, is_training=True):
    data_processed = data.copy()
    date_features = ['F15','F16']
    for date_col in date_features:
        data_processed[date_col] = pd.to_datetime(data_processed[date_col],
errors = 'coerce')
        data_processed[f'{date_col}_year'] = data_processed[date_col].dt.year
        data_processed[f'{date_col}_month'] = data_processed[date_col].dt.year
        data_processed[f'{date_col}_day'] = data_processed[date_col].dt.year

        data_processed[f'{date_col}_days_since_epoch'] =
(data_processed[date_col] - pd.Timestamp('1970-01-01')).dt.days

    if len(date_features) >= 2:
        data_processed['date_diff_days'] = (data_processed[date_features[1]] -
data_processed[date_features[0]]).dt.days

    date_derived_cols = []
    for date_col in date_features:
        date_derived_cols.extend([f'{date_col}_year', f'{date_col}_month',
f'{date_col}_day', f'{date_col}_days_since_epoch'])
    date_derived_cols.append('date_diff_days')
    for col in date_derived_cols:
        if col in data_processed.columns:
            data_processed[col] =
data_processed[col].fillna(data_processed[col].median())
    columns_to_drop = date_features + (['Index'] if is_training else
['Index'])
    data_processed = data_processed.drop(columns=columns_to_drop)

    return data_processed

train_processed = feature_engineer(train_data, is_training=True)
```

```
test_processed = feature_engineer(test_data, is_training=False)

# 3. Model Training
feature_columns = [col for col in train_processed.columns if col != 'C']
X = train_processed[feature_columns]
y = train_processed['C']
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42, stratify=y)

rf = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1)
rf.fit(X_train, y_train)

# 4. Predicting and evaluating
val_predictions = rf.predict(X_val)
val_probabilities = rf.predict_proba(X_val)[:, 1]

# Metrics
accuracy = accuracy_score(y_val, val_predictions)
auc = roc_auc_score(y_val, val_probabilities)
print(f"Validation Accuracy: {accuracy:.4f}")


# Training predictions
train_predictions = rf.predict(X)
test_predictions = rf.predict(test_processed[feature_columns])

# Visualization
fig, axes = plt.subplots(2, 2, figsize=(15, 10))

# ROC Curve
fpr, tpr, _ = roc_curve(y_val, val_probabilities)
axes[0, 0].plot(fpr, tpr, label=f'Random Forest (AUC = {auc:.3f})')
axes[0, 0].plot([0, 1], [0, 1], 'k--', alpha=0.6)
axes[0, 0].set_xlabel('False Positive Rate')
axes[0, 0].set_ylabel('True Positive Rate')
axes[0, 0].set_title('ROC Curve')
axes[0, 0].legend()
axes[0, 0].grid(True, alpha=0.3)

# Confusion Matrix
cm = confusion_matrix(y_val, val_predictions)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ax=axes[0, 1])
axes[0, 1].set_xlabel('Predicted')
axes[0, 1].set_ylabel('Actual')
axes[0, 1].set_title('Confusion Matrix')

# Target Distribution
y.value_counts().plot(kind='bar', ax=axes[1, 0])
axes[1, 1].set_xlabel('Class')
```

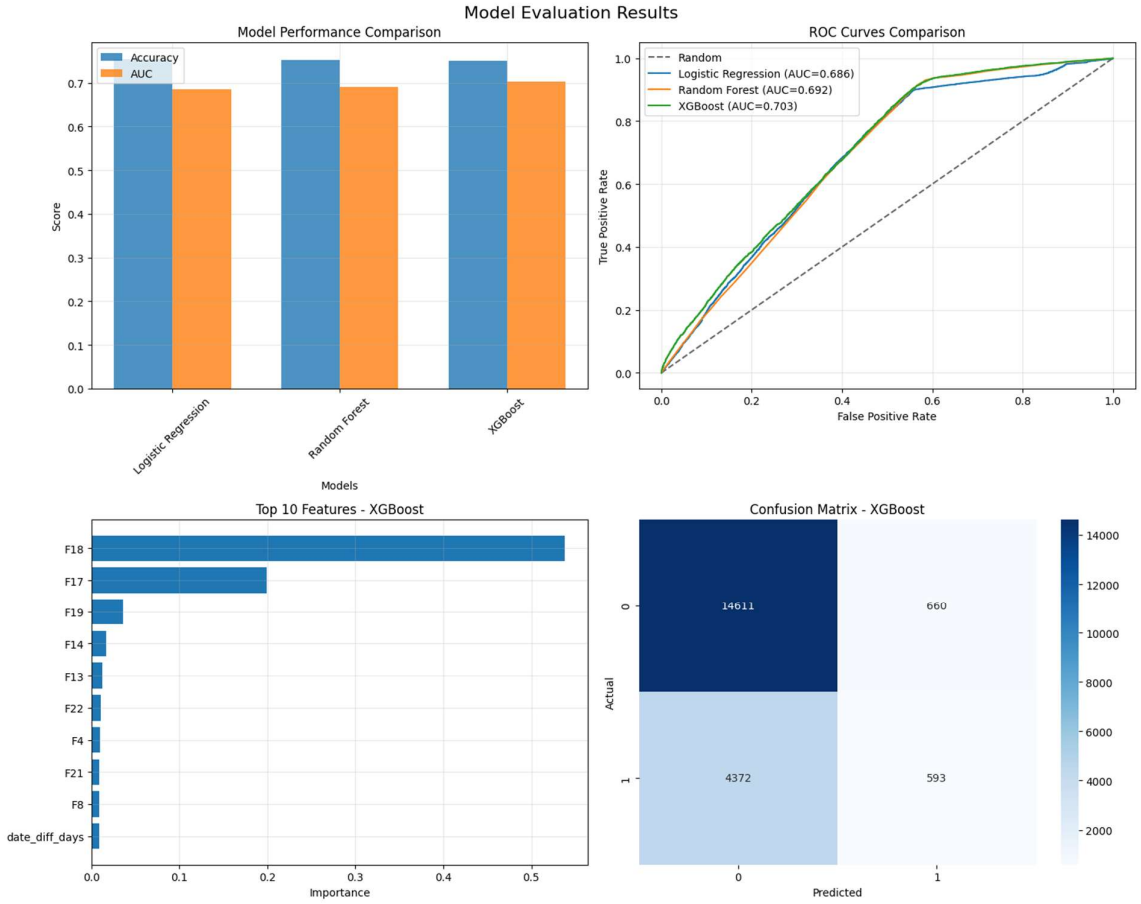
```
axes[1, 1].set_ylabel('Count')
axes[1, 1].set_title('Target Variable Distribution')
axes[1, 1].tick_params(axis='x', rotation=0)
axes[1, 1].axis('off')
plt.tight_layout()
plt.show()

# Saving predictions
train_pred_df = pd.DataFrame({
    'Index': train_data['Index'].values,
    'Class': train_predictions
})
train_pred_df.to_csv('training_predictions.txt', sep='\t', index=False)
test_pred_df = pd.DataFrame({
    'Index': test_data['Index'].values,
    'Class': test_predictions
})
test_pred_df.to_csv('test_predictions.txt', sep='\t', index=False)
```

7. Visualizations

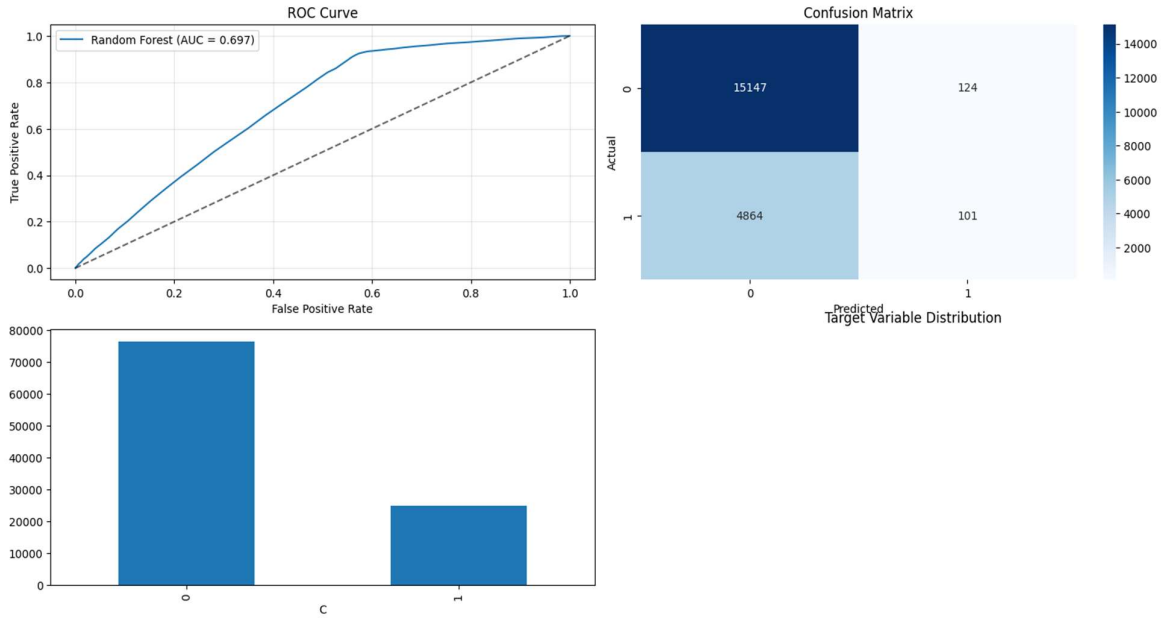
7.1 Basic Statistics and Visualization on Data



7.2 Model Evaluation



7.3 Random Forest Metrics Visualization



8. Conclusion

This project involved a complete end-to-end machine learning workflow — from data exploration and cleaning to model evaluation and selection. The process began with a thorough understanding of the dataset, identifying key features, and applying appropriate preprocessing techniques, including handling missing values and engineering new features such as date differences. Multiple models were trained and evaluated, allowing for a comparison of their performance across various metrics. Through this iterative approach, the most suitable model was chosen based on its balance of accuracy, robustness, and generalization capability. The use of the ROC curve and AUC score provided deeper insight into the model's discriminative power, ensuring that the final choice was supported by both visual and statistical evidence. Overall, the project reinforced the importance of systematic experimentation, critical evaluation, and metric-driven decision-making in building effective predictive models.