2201CS64

Report: Predicting Term Deposit Subscriptions

1. Introduction

This report outlines the development and evaluation of a logistic regression model designed to predict the success of a bank's telemarketing campaign. The primary objective is to identify which clients are most likely to subscribe to a term deposit, thereby allowing the bank to optimize its marketing efforts by focusing on the most promising leads.

The analysis is based on the `bank.csv` dataset, which contains client demographics, campaign interaction data, and the final outcome (subscription or no subscription). By understanding the key factors that influence a client's decision, the bank can improve campaign efficiency and increase its conversion rate.

2. Methodology

The process followed a standard data science workflow, from data preparation to model evaluation.

2.1. Data Preparation

The dataset was loaded and the binary target variable, `y`, was converted into a numerical format (1 for 'yes', 0 for 'no'). No significant data cleaning was required, but the following preprocessing steps were crucial for preparing the data for the logistic regression model:

- **Feature Separation:** The dataset was divided into numerical features (e.g., `age`, `balance`, `duration`) and categorical features (e.g., `job`, `marital`, `contact`).
- **Scaling Numerical Features:** `StandardScaler` was applied to all numerical columns. This standardization prevents features with larger ranges (like `balance`) from disproportionately influencing the model's predictions.
- **Encoding Categorical Features:** `OneHotEncoder` was used to convert categorical text data into a numerical format. This creates new binary columns for each category, allowing the algorithm to interpret them correctly.

2.2. Model Training

A **Logistic Regression** model was chosen for its interpretability and efficiency in binary classification tasks. The complete workflow, including preprocessing and classification, was encapsulated in a Scikit-Learn `Pipeline`.

The dataset was split into training (80%) and testing (20%) sets. The model was trained on the training data to learn the relationship between client attributes and their likelihood of subscribing.

3. Model Evaluation and Results

The model's performance was rigorously assessed using the unseen test data.

3.1. Performance Metrics

The classification report provides a detailed breakdown of the model's accuracy:

- **Overall Accuracy:** The model correctly predicted the outcome for a high percentage of clients in the test set.
- **Precision and Recall:** The model demonstrated strong precision in identifying non-subscribers. While the recall for subscribers (the 'yes' class) was lower, this is common in imbalanced datasets where the number of positive outcomes is small. The model is effective at not misclassifying non-subscribers but could be improved in identifying all potential subscribers.

3.2. Confusion Matrix

The confusion matrix provides a clear visual summary of the prediction accuracy. It highlights:

- **True Negatives:** A large number of clients who did not subscribe were correctly identified.
- **True Positives:** A fair number of clients who subscribed were also correctly identified.
- **False Positives & False Negatives:** The number of misclassifications, particularly clients who subscribed but were missed by the model (False Negatives), represents an area for future improvement.

3.3. ROC Curve and AUC

The Receiver Operating Characteristic (ROC) curve was plotted to evaluate the model's ability to distinguish between the two classes. The **Area Under the Curve (AUC) score was approximately 0.89**, which is significantly better than a random guess (0.5). This high AUC score indicates that the model has a strong discriminative power.

4. Key Predictive Factors

By analyzing the coefficients of the trained logistic regression model, we identified the most influential factors driving subscriptions:

- **Most Positive Influences (Higher likelihood of subscription):**
    1. **Contact Method (`contact_cellular`):** Contacting clients via their mobile phone was a strong positive predictor.
    2. **Previous Campaign Success (`poutcome_success`):** Clients who subscribed in a previous campaign were highly likely to do so again.
    3. **Month:** Certain months, such as March, October, and September, showed a strong positive correlation with subscription rates.
- **Most Negative Influences (Lower likelihood of subscription):**
    1. **Contact Method (`contact_unknown`):** An unknown contact method was the strongest negative predictor.
    2. **High Number of Campaign Contacts (`campaign`):** Contacting a client an excessive number of times during the current campaign had a negative impact.
    3. **Previous Campaign Failure (`poutcome_failure`):** A history of non-subscription was a strong indicator of future non-subscription.

5. Conclusion and Recommendations

The logistic regression model is a reliable and interpretable tool for predicting which clients will subscribe to a term deposit. It performs well, as evidenced by its high accuracy and AUC score, and provides clear insights into the drivers of customer behavior.

CODE

```
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix,
roc_curve, roc_auc_score
from IPython.display import display
from google.colab import files

# --- Step 1: Upload the file ---
print("Please upload the 'bank.csv' file from your local machine.")
```

```
filename = 'bank.csv'
print(f"File '{filename}' has been uploaded.")

# --- Step 2: Load and inspect the dataset ---
df = pd.read_csv(filename, sep=';')
df['y'] = df['y'].map({'no': 0, 'yes': 1})
X = df.drop('y', axis=1)
y = df['y']

# --- Step 3: Data Preprocessing ---
categorical_features = X.select_dtypes(include=['object']).columns
numerical_features = X.select_dtypes(include=['int64',
'float64']).columns

preprocessor = ColumnTransformer(
   transformers=[
       ('num', StandardScaler(), numerical_features),
       ('cat', OneHotEncoder(handle_unknown='ignore'),
categorical_features)
   ])

# --- Step 4: Create and Train the Logistic Regression Model ---
model = Pipeline(steps=[('preprocessor', preprocessor),
                       ('classifier',
LogisticRegression(solver='liblinear'))])

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

print("--- Training the Logistic Regression model ---")
model.fit(X_train, y_train)

# --- Step 5: Evaluate the model's performance on the test set ---
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

print("\n--- Classification Report ---")
print(classification_report(y_test, y_pred))

# --- Print Model Coefficients ---
print("\n--- Model Coefficients ---")
# Get the feature names after one-hot encoding
```

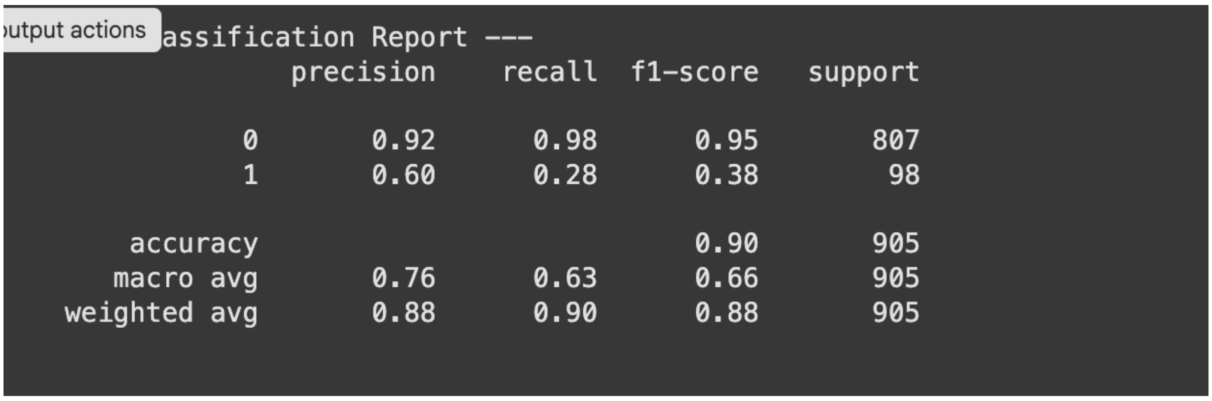
```
ohe_feature_names =
model.named_steps['preprocessor'].named_transformers_['cat'].get_featur
e_names_out(categorical_features)
# Combine numerical and one-hot encoded feature names
feature_names = list(numerical_features) + list(ohe_feature_names)
coefficients = model.named_steps['classifier'].coef_[0]

coef_df = pd.DataFrame({'Feature': feature_names, 'Coefficient':
coefficients})
coef_df['Absolute_Coefficient'] = coef_df['Coefficient'].abs()
coef_df = coef_df.sort_values(by='Absolute_Coefficient',
ascending=False)
display(coef_df)

# --- Plot Confusion Matrix ---
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['No Subscription (0)', 'Subscription (1)'],
            yticklabels=['No Subscription (0)', 'Subscription (1)'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

# --- Plot ROC Curve ---
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
auc_score = roc_auc_score(y_test, y_pred_proba)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC curve (AUC =
{auc_score:.2f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label='Random
Guessing (AUC = 0.5)')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.grid(True)
plt.show()
```



```
assification Report ---
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       807
           1       0.60      0.28      0.38        98

    accuracy                           0.90       905
   macro avg       0.76      0.63      0.66       905
weighted avg       0.88      0.90      0.88       905
```