Leela Sowmya Jandhyala (u1472955)
CS 5350/6350 Machine Learning Fall 2024

Mid-Term Report for Kaggle Project: Income Level Prediction for Mortgage

Introduction

For this project, the goal is to predict whether a resident's yearly income exceeds 50K based on several features, both demographic and financial, using machine learning models. The project uses a dataset derived from census data, and submissions are evaluated using the Area Under the ROC Curve (AUC), which measures the performance of binary classifiers.
In this report, I will outline the preprocessing steps applied to the dataset, the models used for training, and the initial results obtained. I will also highlight the planned steps for improving the model's performance in the next phase of the project.

Data Pre-processing

The dataset contains both continuous i.e. numerical and categorical attributes. Some attributes have missing values represented as question marks (?). The preprocessing pipeline was designed to clean and transform both categorical and numerical columns to make the dataset suitable for machine learning models.

Handling Missing Values: First, the occurrences of ? in the dataset were identified and replaced with NaN. This allowed us to impute missing values using appropriate strategies for numerical and categorical features.

```
# Replace '?' with NaN to handle missing values
train_df.replace('?', pd.NA, inplace=True)
test_df.replace('?', pd.NA, inplace=True)
```

Numerical Features: For continuous i.e. numerical columns ['age', 'fnlwgt', 'education.num', 'capital.gain', 'capital.loss', 'hours.per.week'], missing values were imputed using the mean strategy, and then the features were scaled using StandardScaler to standardize them.

```
# handling numerical columns
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])
```

Categorical features: Categorical features were split into low and high cardinality categories based on the number of unique values, with a threshold of 15. If a column had more than 15 unique values, it was considered high cardinality; otherwise, it was considered low cardinality. Low cardinality features were encoded using OneHotEncoder to create binary columns, while high cardinality features were encoded using OrdinalEncoder to assign integer values to different categories. This approach ensured efficient handling of features like "native.country" and "education," which have many unique values.

```
# Handling categorical columns
categorical_transformer_low = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

categorical_transformer_high = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
       ('ordinal',  OrdinalEncoder(handle_unknown='use_encoded_value',
unknown_value=-1))
])
```

I then combined the above preprocessing steps by using a ColumnTransformer, which applied the necessary transformations to the appropriate columns.

Model Building

After preprocessing the data, I built two baseline models: Logistic Regression and Decision Tree. These models provided a good starting point to understand initial performance levels for binary classification on this dataset.

Logistic Regression Model: Logistic Regression was chosen for its simplicity and effectiveness. After training, it achieved an AUC score of 0.9075 on the training set, indicating it captures relevant patterns in the data quite well.

```
# Logistic Regression Model
logreg = LogisticRegression(random_state=0)

# Training Logistic Regression
logreg.fit(X_train_transformed, y_train)

# Predicting
y_train_pred_logreg = logreg.predict_proba(X_train_transformed)[:, 1]
# Probability for class 1 (income > 50K)

# Evaluate Logistic Regression Model using Area Under ROC (AUC) curve
logreg_auc = roc_auc_score(y_train, y_train_pred_logreg)

#  Generate  predictions  on  the  test  data  using  Logistic  Regression
Model
y_test_pred_logreg = logreg.predict_proba(X_test_transformed)[:, 1]
```

Decision Tree Model: I used a Decision Tree with a maximum depth of 10 and minimum samples per leaf of 5 to prevent overfitting. This model achieved an AUC score of 0.9273 on the training set, indicating that non-linear relations in the data might improve predictive performance.

```
# Decision Tree Model
decision_tree  =  DecisionTreeClassifier(random_state=0,  max_depth=10,
min_samples_leaf=5)

# Train Decision Tree
decision_tree.fit(X_train_transformed, y_train)

# Predicting
y_train_pred_tree                                                    =
decision_tree.predict_proba(X_train_transformed)[:, 1]

# Evaluate Decision Tree Model using Area Under ROC (AUC) curve
tree_auc = roc_auc_score(y_train, y_train_pred_tree)

# Generate predictions on the test data using Decision Tree Model
y_test_pred_tree = decision_tree.predict_proba(X_test_transformed)[:,
1]
```

Submissions
After training the models, I generated predictions on the test set and created submission files for both Logistic Regression and Decision Tree. These submissions were in the required format, with each prediction representing the probability of a person earning more than 50K.

Future Work
Moving forward, I plan to build on the foundation that has already been established with these baseline models by exploring more sophisticated techniques to improve the model's predictions and increase the AUC score. I will be focusing on the following areas:

Feature Engineering: I plan to create new features by combining or transforming existing ones, such as generating interaction terms or deriving new insights from the data. This will help capture additional patterns.
Advanced Model: Building on the insights from the baseline models, I will experiment with more complex models such as Random Forest and Gradient Boosting. These models are known for their ability to handle complex relationships and this may result in better predictions.
Hyperparameter Tuning: To improve model performance, I will use grid search to fine-tune parameters, such as maximum tree depth, minimum samples per leaf, and learning rate for ensemble models. This tuning process will help optimize the model's complexity and accuracy, enabling it to better capture underlying patterns in the data.

Conclusion
In this report, I have outlined the preprocessing steps taken to clean and transform the dataset, as well as the baseline models built for predicting income level. By evaluating the initial performance of Logistic Regression and Decision Tree models, I have established a solid foundation for future steps. The next phase of this project will focus on feature engineering, applying more advanced models, and optimizing performance through hyperparameter tuning. These improvements will help increase the model's overall accuracy and efficiency.