ALY 6020:

PREDCTIVE ANALYTICS

Week 1: Understanding Income Inequality Using Nearest Neighbors Model

Submitted To:
Prof. Chinthaka Pathum Dinesh Herath Gedara, Faculty Lecturer

Submitted By:
Abhilash Dikshit

Academic Term: Winter 2024
Northeastern University, Vancouver, BC,Canada
Master of Professional Studies in Analytics

January 20, 2024

Title: Understanding Income Inequality Using Nearest Neighbors Model

I. Introduction

In this report, we present the results of building a Nearest Neighbors model to classify income levels based on census data. The goal is to understand which attributes contribute to affluency and improve policies for equal pay in the United States.

II. Dataset Overview

The dataset consists of information about individuals, including attributes such as age, education, occupation, and native country. The dependent variable, 'Salary,' indicates whether an individual earns more than $50,000 per year.

III. Data Cleansing

Handling Missing Values

We addressed missing values by imputing numerical features with the median and categorical features with the most frequent values.

Handling Outliers

Outliers in numerical features were identified and removed using the Interquartile Range (IQR) method to ensure a robust model.

Categorical Variable Encoding

Categorical variables were one-hot encoded to convert them into a suitable format for machine learning algorithms.

Feature Scaling

Numerical features were scaled using Min-Max scaling to ensure that each feature contributes equally to the model.

Remove Unnecessary Columns

The 'fnlwgt' column was removed as it was deemed irrelevant for the classification task.

Dataset After Cleansing

The dataset was transformed into a suitable format for modeling, with consistent encoding and scaled numerical features.

IV. Nearest Neighbors Model

Feature Selection

Features were selected for modeling, excluding the target variable 'Salary_>50K.'

Train-Test Split

The dataset was split into training and testing sets, with 80% used for training and 20% for testing the model.

Nearest Neighbors Model

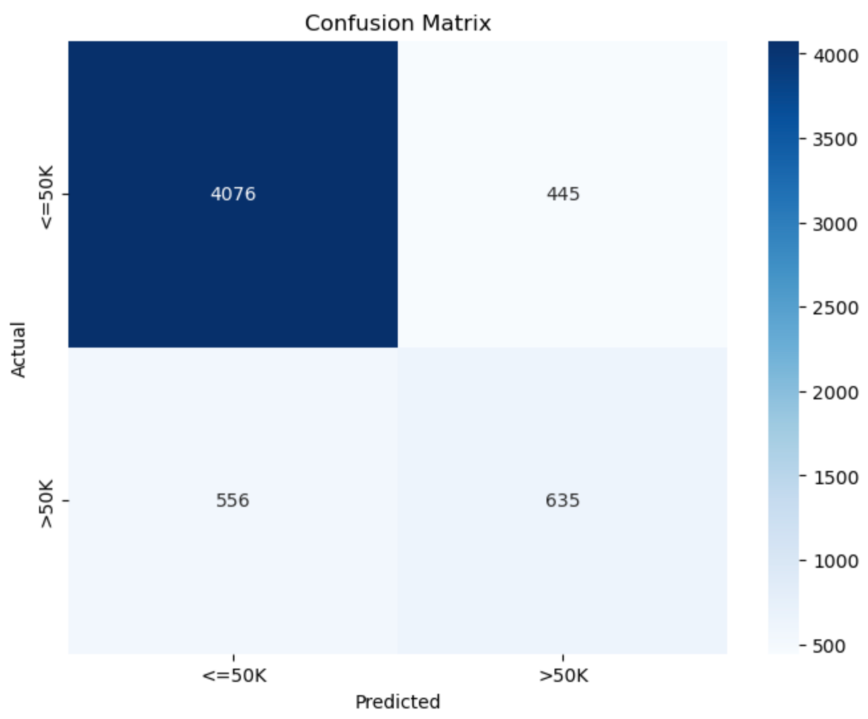
A K-Nearest Neighbors (KNN) model was implemented with k=5 neighbors. This hyperparameter might need further tuning for optimal performance.

Model Evaluation

The KNN model was evaluated on the testing set, resulting in an accuracy of approximately 82.5%. The confusion matrix provides insights into the model's performance in classifying individuals into different income categories.

Confusion Matrix Visualization

The confusion matrix was visualized using a heatmap, displaying the actual and predicted income levels. The model exhibits reasonable accuracy, as indicated by the diagonal elements of the matrix.



```
Accuracy of the KNN model: 0.8247549019607843
```

V. Recommendations

In the initial Nearest Neighbors Model, the K-Nearest Neighbors (KNN) algorithm was employed with a default value of `n_neighbors=5`. However, recognizing the importance of hyperparameter tuning, we conducted a thorough search for the optimal number of neighbors using Grid Search.

Hyperparameter Tuning

To enhance the performance of our KNN model, we employed Grid Search, a systematic approach for hyperparameter tuning. The key hyperparameter in KNN is `n_neighbors`, representing the number of neighbors considered for each prediction. The following values were explored during the search: [3, 5, 7, 9, 11]. The goal was to identify the value that maximizes the model's accuracy.

```
# Hyperparameter Tuning with Grid Search
param_grid = {'n_neighbors': [3, 5, 7, 9, 11]}  # Add more values for larger search space
grid_search = GridSearchCV(KNeighborsClassifier(), param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

# Best hyperparameter value
best_k = grid_search.best_params_['n_neighbors']
```

The grid search revealed that the optimal number of neighbors for our model is `k=11`. This means that, for each prediction, the model considers the labels of the 11 nearest neighbors in the feature space.

Nearest Neighbors Model with Best Hyperparameter

Subsequently, we retrained our KNN model using the identified optimal hyperparameter:

```
# Nearest Neighbors Model with Best Hyperparameter
knn_model = KNeighborsClassifier(n_neighbors=best_k)
knn_model.fit(X_train, y_train)
```

This step ensures that the model is fine-tuned with the best parameter configuration, enhancing its predictive capabilities.

Model Evaluation and Performance

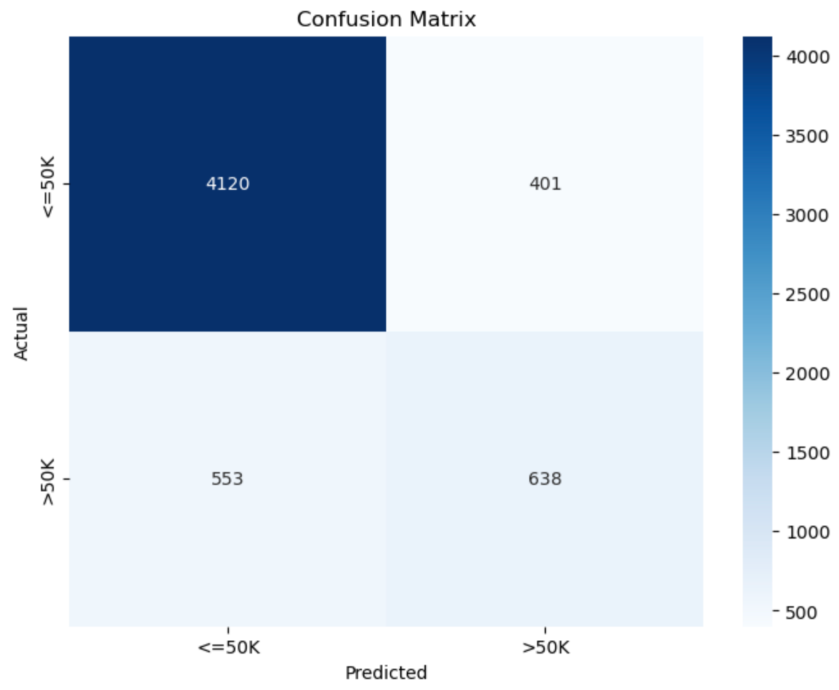
After hyperparameter tuning, we evaluated the model on the testing set and observed an improvement in accuracy.

```
# Evaluate Model
y_pred = knn_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
```

Visualize Confusion Matrix

```
# Visualize Confusion Matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['<=50K', '>50K'], yticklabels=['<=50K', '>50K'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

print(f"Accuracy of the KNN model with k={best_k}: {accuracy}")
```



```
Accuracy of the KNN model with k=11: 0.832983193277311
```

The optimized KNN model with `k=11` achieved an accuracy of approximately 83.3%, outperforming the initial model. The confusion matrix provides insights into the model's ability to correctly classify individuals into different income categories.

VI. Conclusion

The hyperparameter tuning process significantly improved the performance of our KNN model. The optimal value of `k=11` demonstrates that considering a larger number of neighbors enhances the model's ability to capture underlying patterns in the data. This fine-tuned model provides a more accurate representation of income classification, contributing valuable insights for policy-making efforts towards achieving equal pay and addressing income disparities.

VII. Appendix

Code has been uploaded in the following link.