Logo

Description automatically generated

**College of Professional Studies**

**Northeastern University San Jose**

**MPS Analytics**

**Course: ALY6020 – Predictive Analytics**

**Module 1 Project:** Understanding Income Inequality

**Submitted on:**

Sept 28th , 2023

**Submitted to:**  **Submitted by:**

Professor: BEHZAD AHMADI BHAGYASHRI KADAM

**INTRODUCTION**

In a diverse and dynamic society like the United States, understanding the socio-economic factors influencing citizens' income levels is crucial for policy formulation and societal advancement. The U.S. Census Bureau collects extensive data on citizens, capturing various attributes such as occupation, education, gender, and race. Analyzing this data provides insights into the income disparities among different groups and aids in identifying areas requiring intervention to ensure equal opportunities and fair income distribution.

The primary objective of this project is to build a k-Nearest Neighbors (k-NN) model that can accurately classify U.S. citizens into low and high-income categories based on their attributes, using the given census data.

**Understanding the Dataset**

The given dataset comprises a total of **48842 records and 15 columns/attributes** each representing an individual citizen and detailing their respective attributes such as age, education, marital status etc.

The target variable for this dataset is salary which is categorized as <=50K and >=50K

Below are the datatypes of the attributes:

A white background with black text

Description automatically generated

**Numerical Attributes:** Age, Fnlwgt, Education-num, Capital-gain, Capital-loss, Hours-per-week.

**Categorical Attributes:** Workclass, Education, Marital-status, Occupation, Relationship, Race, Sex, Native-country, Salary.

**PART 1 : DATA CLEANING**

* **Checking the number of missing values for each Attribute in the dataset**

A screenshot of a computer

Description automatically generated

From the above output, we can observe that there are no missing values for any of the attributes.

**DESCRIPTIVE CHARACTERISTICS OF THE DATASET**

**A screenshot of a graph

Description automatically generated**

The above displays the statistical information for several numerical columns in the DataFrame. The columns included are "Age", "fnlwgt", "education-num", "capital-gain", "capital-loss" and “hours-per-week”. The statistical information provides insights such as count, mean, standard deviation, minimum, maximum, and quartiles for each column, giving an overview of the data distribution and variability.

The key statistical information for the numerical columns in the Data frame is as follows:

- The attributes Capital-gain and Capital-loss have a mean value relatively close to zero, with a majority of their data points being zero, indicating that most citizens did not have capital gains or losses.

- The Age attribute has a wide range, from 17 to 90, with a mean age of approximately 38.64 years.

- The Hours-per-week attribute, representing the working hours per week, has a mean value of approximately 40.42, aligning with the standard full-time working hours, but it also has a wide range, indicating varied working hours among the citizens

**DATA VISUALIZATIONS**

In the EDA , we will start by understanding the distribution of numerical attributes using histogram as below :

A graph of a graph

Description automatically generatedA graph of a loss

Description automatically generated with medium confidenceAnalyzing multiple histograms for the numerical variables in the dataset provided us insights into their distributions and patterns.

Age attribute shows a right-skewed distribution which indicates that there are more younger individuals.

Capital Gain & Capital loss have lot of entries at 0 representing no capital gain/loss , leading to a right-skewed distribution

Lets further check the distribution of categorical attributes :

A bar graph with different colored squares

Description automatically generatedA graph with colorful bars

Description automatically generated

A bar graph with numbers and text

Description automatically generated with medium confidence

Below are some insights we can observe from above bar plots :

* Most of the people are working in the private sector work class.
* High school degree is the most common degree held by folks followed by some college degree and bachelors.
* Also most of the population is of white race ,there are more males than females in the given data and most of the population is from native country as United states followed by Mexico.
* Lastly More than half of the population is having salary <=50K

**Correlation Map**

A screenshot of a graph

Description automatically generated

The above correlation map helps us to understand the relationship between the numerical attributes of the dataset. The correlation between education-num and hours-per-week is the highest followed by capital-gain and education-num.

**Part 2 : Model Building & Analysis**

Since we need to build a KNN model , we will begin with encoding the categorial variables as we need to provide a numerical input to the machine learning model and one-hot encoding enables the representation of categorical data in to numerical format (binary vector) for making better predictions.

Additionally , if we do label encoding and simply convert the categorical data to integers , the model might misunderstand the data to be in some kind of order, which we want to avoid for nominal categorical variables.

The python library pandas provides a function called get\_dummies to enable one-hot encoding

A screenshot of a computer

Description automatically generated

*One-hot encoding*

Further, we have performed feature-scaling which is crucial for normalizing the feature set to ensure uniform impact and is especially important for distance-based algorithms like KNN. Further , feature scaling also ensures that no particular feature dominates the model due to its scale and that all features contribute equally to the distance computation.

I have used the scaling method of standardization to only the numerical attributes and not the One-hot encoded features as shown below. As standardizing the one-hot encoded features would mean assigning a distribution to the categorical features which we do not want.

The numerical features are now centered on the mean with a unit standard deviation.

A screenshot of a computer

Description automatically generated

*Feature Scaling*

**Building the KNN Model :**

Before building the model , I have divided the data in to target variable (Salary) and the rest if the features and then split the data in to training set (80%) & test set (20%)

A screenshot of a computer program

Description automatically generated

*Split the data*

Further, to find the **optimal value of K** , I have used the grid search cross validation with the range of (1,31) inclusive. GridsearchCV will try each of these values for ‘K’ to find the one that results in the best model performance.

The number of folds used for cross validation is 5 means that the training data will be split in to 5 folds and 5 iterations of training & validation will be performed, each time with different fold used as the validation set. The optimal value of K is 22.

A screenshot of a computer code

Description automatically generated

Further , I have trained the model with the optimal value of k ( best k) found using GridsearchCV and is assigned to knn\_optimal. The model knn\_optimal is then trained on the training data and this trained model is used to make predictions on the test data and the predicted values are stored in y\_pred.

Lastly , have calculated the model’s accuracy using the ‘accuracy score’ function , comparing the predicted values (‘y\_pred’) to the true values (‘y\_test’). The model’s overall accuracy is 84%

A screenshot of a computer program

Description automatically generated

Further, classification report is generated using the classification report function providing detailed performance metrics like precision, recall , f1-score and support for each class.

A screenshot of a computer

Description automatically generated

From the above classification report , we can understand that the model has high precision & recall for class 0 indicating good model performance for citizens with a salary less than equal to 50K (low income).

However, the precision is relatively high , recall is lower for citizens with a salary greater than 50K(High Income)

**Accuracy plot for different values of K:**

A graph with a line going up

Description automatically generated

**Confusion Matrix :**

A screenshot of a graph

Description automatically generated

From the above confusion matrix, we can understand that the model correctly predicted 6876 instances where the salary is <=50K and incorrectly predicted 538 instances as having salary >50K which means there is room for improvement in reducing the number of false alarms.

The model incorrectly predicted 995 instances as having a salary <=50K when the actual salary is >50K which suggests that the model is missing out on identifying a considerable number of individuals with salaries >50K.

Lastly the model correctly predicted 1360 instances where salary >50K

**Conclusion**

The Nearest Neighbors model was able to classify individuals into high and low-income categories with an overall accuracy of 84%. The model excelled in identifying individuals with salaries ≤50K, with a high precision of 87.36% and recall of 92.74%.

The model performs well in classifying citizens with a salary of <=50K , however the model experienced challenges in accurately classifying citizens with salaries >50K, reflected by a lower recall of 57.75% for this class, indicating a substantial number of high-income individuals were missed. Further, the discrepancy is recall between the two classes indicates an imbalance in the dataset.

The presence of 545 False Positives and 995 False Negatives highlights the necessity for model refinement to reduce misclassifications and improve reliability.

We would need to further refine the model to address the identified areas of improvement, particularly focusing on enhancing the recall for high-income classifications and reducing the number of false alarms.

Exploring alternative classification algorithms, feature engineering, and addressing class imbalance can contribute to model enhancement.

**References**

1. <https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb#:~:text=The%20optimal%20K%20value%20usually,be%20aware%20of%20the%20outliers>.
2. <https://towardsdatascience.com/preprocessing-encode-and-knn-impute-all-categorical-features-fast-b05f50b4dfaa>

**Appendix**

|  |
| --- |
| import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sns  #Loading data#  df = pd.read\_csv("/Users/bhagyashrikadam/Documents/NEU\_ASSIGNMENTS/ALY6020/Module1/Module1 Project/adult-all.csv")  df.shape  print(df.dtypes)  df.head(10)  ## PART1 ##  ## Data Cleaning ##  # Checking the number of missing values ##  missing\_values = df.isnull().sum()  print(missing\_values)  # Descriptive Statistics #  df.describe()  ######## EDA ########  # Extracting numerical features from the dataset  numerical\_features = df.select\_dtypes(include=['int64', 'float64']).columns.tolist()  # Calculating the number of rows required for the subplots  num\_rows = -(-len(numerical\_features) // 3) # Ceiling division  # Setting up the figure and axis  fig, axs = plt.subplots(num\_rows, 3, figsize=(15, 5 \* num\_rows))  axs = axs.ravel() # Flattening the axis array  # Plotting histograms for each numerical feature  for i, feature in enumerate(numerical\_features):  sns.histplot(df[feature], kde=True, bins=30, ax=axs[i])  axs[i].set\_title(f'Histogram for {feature}')  axs[i].set\_xlabel(feature)  axs[i].set\_ylabel('Frequency')  # Removing empty subplots  for i in range(len(numerical\_features), num\_rows \* 3):  fig.delaxes(axs[i])  # Displaying the plots  plt.tight\_layout()  plt.show()  # Extracting categorical features from the dataset  categorical\_features = df.select\_dtypes(include=['object']).columns.tolist()  # Calculating the number of rows required for the subplots  num\_rows = -(-len(categorical\_features) // 3) # Ceiling division  # Setting up the figure and axis  fig, axs = plt.subplots(num\_rows, 3, figsize=(15, 5 \* num\_rows))  axs = axs.ravel() # Flattening the axis array  # Plotting bar plots for each categorical feature  for i, feature in enumerate(categorical\_features):  sns.countplot(y=df[feature], order=df[feature].value\_counts().index, ax=axs[i])  axs[i].set\_title(f'Bar Plot for {feature}')  axs[i].set\_xlabel('Count')  axs[i].set\_ylabel(feature)  # Removing empty subplots  for i in range(len(categorical\_features), num\_rows \* 3):  fig.delaxes(axs[i])  # Displaying the plots  plt.tight\_layout()  plt.show()  # 3. Correlation Heatmap for Numerical Features  plt.figure(figsize=(10, 8))  sns.heatmap(df[numerical\_features].corr(), annot=True, cmap='coolwarm', fmt='.2f')  plt.title('Correlation Heatmap for Numerical Features')  plt.show() |
| # Set the aesthetic style of the plots  sns.set\_style("whitegrid")  # Create a figure and axis object  plt.figure(figsize=(10,6))  # Plot the distribution of the 'Salary' variable  sns.histplot(df['salary'], kde=True, bins=30)  # Set the title and labels of the plot  plt.title('Distribution of Salary')  plt.xlabel('Salary')  plt.ylabel('Frequency')  # Show the plot  plt.show()  ## PART 2 : MODEL BUILDING & ANALYSIS ###  #Encode Categorical Variables  # Identify categorical variables in the dataset  categorical\_features = df.select\_dtypes(include=['object']).columns  #We will use One-Hot Encoding for categorical variables  data\_encoded = pd.get\_dummies(df, columns=categorical\_features, drop\_first=True)  # Display the first few rows of the encoded dataframe  data\_encoded.head()  # Feature Scaling  from sklearn.preprocessing import StandardScaler  # Identify numerical features to scale  numerical\_features\_to\_scale = data\_encoded.select\_dtypes(include=['int64', 'float64']).columns  # Initialize the Standard Scaler  scaler = StandardScaler()  # Scale the numerical featuresx  data\_encoded[numerical\_features\_to\_scale] = scaler.fit\_transform(data\_encoded[numerical\_features\_to\_scale])  # Display the first few rows of the scaled dataframe  data\_encoded.head()  # Build a Nearest Neighbors Model #  from sklearn.model\_selection import train\_test\_split  from sklearn.neighbors import KNeighborsClassifier  from sklearn.model\_selection import GridSearchCV  from sklearn.metrics import accuracy\_score, classification\_report  # 1. Split the Data  # Define the feature set X and the target variable y  X = data\_encoded.drop('salary\_>50K', axis=1)  y = data\_encoded['salary\_>50K']  # Split the 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)  # 2. Select K  # Use GridSearchCV to find the optimal value of K  param\_grid = {'n\_neighbors': range(1, 31)}  knn = KNeighborsClassifier()  knn\_cv = GridSearchCV(knn, param\_grid, cv=5)  knn\_cv.fit(X\_train, y\_train)  # Display the optimal value of K  best\_k = knn\_cv.best\_params\_['n\_neighbors']  best\_k  # 3. Train the Model  from sklearn.metrics import classification\_report  # Initialize the K-Nearest Neighbors model with the optimal value of K  knn\_optimal = KNeighborsClassifier(n\_neighbors=best\_k)  # Train the model on the training data  knn\_optimal.fit(X\_train, y\_train)  # 4. Evaluate the Model  # Predict the target variable on the testing data  y\_pred = knn\_optimal.predict(X\_test)  # Calculate the accuracy of the model  accuracy = accuracy\_score(y\_test, y\_pred)  # Display the accuracy  Accuracy  # Generate the classification report  report = classification\_report(y\_test, y\_pred, output\_dict=True)  # Convert the report to a DataFrame  report\_df = pd.DataFrame(report).transpose()  # Display the report in tabular format  report\_df  ## Accuracy Plot for Different values of K ##  from sklearn.metrics import confusion\_matrix  from sklearn.model\_selection import cross\_val\_score  # 1. Accuracy Plot for different values of K during cross-validation  k\_values = range(1, 22) # Considering K values from 1 to 20  cross\_val\_accuracies = []  for k in k\_values:  knn\_model = KNeighborsClassifier(n\_neighbors=k)  scores = cross\_val\_score(knn\_model, X\_train, y\_train, cv=5, scoring='accuracy')  cross\_val\_accuracies.append(scores.mean())  plt.figure(figsize=(10, 6))  plt.plot(k\_values, cross\_val\_accuracies, marker='o')  plt.title('Accuracy Plot for Different Values of K')  plt.xlabel('Value of K')  plt.ylabel('Cross-Validated Accuracy')  plt.show()  ## Confusion Matrix ##  from sklearn.metrics import confusion\_matrix  import seaborn as sns  import matplotlib.pyplot as plt  # Compute the confusion matrix  conf\_matrix = confusion\_matrix(y\_test, y\_pred)  # Create a heatmap for the confusion matrix  plt.figure(figsize=(8, 6))  sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues',  xticklabels=['Predicted 0 (<=50K)', 'Predicted 1 (>50K)'],  yticklabels=['Actual 0 (<=50K)', 'Actual 1 (>50K)'])  plt.title('Confusion Matrix displaying TP, TN, FP, FN')  plt.xlabel('Predicted Label')  plt.ylabel('True Label')  plt.show() |