Income Prediction based on Census Data

Group 12

Hari Narayana Gurram
Sai Deepthi Gali
Sahil Sushil Mahajan
Prabhat Chanda
Ruchirkanth Gandikota

gurram.har@northeastern.edu
gali.s@northeastern.edu
sa.mahajan@northeastern.edu
chanda.p@northeastern.edu
gandikota.r@northeastern.edu

Percentage of Effort Contributed by Student 1: **20%**
Percentage of Effort Contributed by Student 2: **20%**
Percentage of Effort Contributed by Student 3: **20%**
Percentage of Effort Contributed by Student 4: **20%**
Percentage of Effort Contributed by Student 5: **20%**

Signature of Student 1: ***Hari Narayana Gurram***
Signature of Student 2: ***Sai Deepthi Gali***
Signature of Student 3: ***Sahil Sushil Mahajan***
Signature of Student 4: ***Prabhat Chanda***
Signature of Student 5: ***Ruchirkanth Gandikota***

Submission Date: **12-08-2023**

Table of Contents

Abstract

In a time of increasing socio-economic disparities and limited government resources, accurately identifying individuals who require assistance is crucial for effective policy making. The “Census income” dataset from the UCI Machine Learning Repository serves as a valuable resource for addressing the issue. This dataset contains a diverse range of demographic and employment related features such as age, education level, occupation, workclass, marital status, and more, providing a snapshot of an individual's socio economic status.

Leveraging machine learning techniques and data analysis, the study aims to construct predictive models capable of accurately categorizing individuals into income brackets. By employing rigorous preprocessing, feature selection, and model evaluation methodologies, the analysis seeks to uncover valuable socio-economic insights regarding the determinants of income levels within the studied population. The derived findings are expected to contribute to a better understanding of socio-economic disparities and assist in informed decision-making processes. This model can aid government agencies in allocating resources efficiently, ensuring that those in need receive the appropriate support while optimizing budget utilization

Problem Definition

The objective of this project is to build a classification model that predicts whether an individual’s income exceeds a threshold of $50,000 per year, based on their demographic and employment related attributes. The problem can be formulated as a binary classification task, where the target variable is binary, indicating whether the individual falls into the “high-income > $50K/yr” or “low-income <= $50K/yr” category.

This predictive tool will be instrumental in streamlining government assistance programs, enabling them to target the right individuals effectively. Additionally, it will contribute to the reduction of resource wastage, as resources can be directed towards those who require them most urgently, aligning with the government’s objective of improving overall social welfare.

Data Sources

The data is taken from the UCI Machine Learning Repository.

https://archive.ics.uci.edu/dataset/20/census+income

Data Description

The Census Income dataset provides a collection of demographic and employment related attributes for individuals. Each record in the dataset represents an individual, and the goal is to predict whether their income exceeds a threshold of $50,000 per year.

It consists of 48,842 records and 15 features in total along with missing values in a few attributes.

1. Age : Age of the individual (numeric).
2. Workclass : The type of employment (categorical: e.g.,Private, Self-emp-not-inc, Never-worked etc).
3. Fnlwgt : A weighting factor used to make the sample more representative of the total population (numeric).
4. Education : Highest level of education achieved (categorical: e.g., Bachelors, Some-college, 11th, etc).
5. Education - num : Numeric representation of education level (numeric).
6. Marital - status : Marital status of the individual (categorical: e.g., Married - civ - spouse, Divorced, Never - married etc).
7. Occupation : Type of occupation (categorical: e.g., Tech - support, Craft - repair, Sales etc).
8. Relationship : Relationship status (categorical: e.g., Wife, Own - child, Husband etc).
9. Race : Ethnicity of the individual (categorical: e.g., White, Asian - Pac - Islander, Black etc).
10. Sex : Gender of the individual (categorical: e.g., Male or Female).
11. Capital - gain : Capital gains reported (numeric).
12. Capital - loss : Capital losses reported (numeric).
13. Hour - per - week : Hours worked per week (numeric).
14. Native - Country : Country of origin (categorical: e.g., United - States, Cambodia, England etc).

Target variable:

- Income : Binary variable indicating whether the individual's income exceeds a specific threshold (<=50K or >50K).

Exploratory Data Analysis

1) Data Exploration and Cleaning:

Processing unrefined data and converting it into a readily usable format for examination and investigation constitutes the tasks of data cleansing and exploration.

Firstly, we begin by loading the dataset, after which we retrieve the initial five rows to gain an understanding of the columns and their specific data types

```
df.head(5)
```

	Age	Workclass	Fnlwgt	Education	Education - num	Marital - status	Occupation	Relationship	Race	Sex	Capital - gain	Capital - loss	Hour - per - week	Native - Country	Income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

As our goal is to enhance the data for easier analysis, it's essential to ensure the absence of duplicate entries or empty values. These could negatively impact reporting accuracy and lead to distorted metrics. Upon examination, we observed no duplicates or null values that require elimination

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30162 entries, 0 to 30161
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Age                 30162 non-null  int64 
 1   Workclass           30162 non-null  object
 2   Fnlwgt              30162 non-null  int64 
 3   Education           30162 non-null  object
 4   Education - num     30162 non-null  int64 
 5   Marital - status    30162 non-null  object
 6   Occupation          30162 non-null  object
 7   Relationship        30162 non-null  object
 8   Race                30162 non-null  object
 9   Sex                 30162 non-null  object
 10  Capital - gain      30162 non-null  int64 
 11  Capital - loss      30162 non-null  int64 
 12  Hour - per - week   30162 non-null  int64 
 13  Native - Country    30162 non-null  object
 14  Income              30162 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.5+ MB
```

However, The dataset contains instances where values are represented by the '?' symbol, indicating missing or unknown information. Therefore, Rows containing the '?' symbol are removed or dropped from the dataset to ensure data integrity and analysis accuracy

```
df.eq(" ?").sum()
```

```
Age                     0
Workclass            1616
Education               0
Marital - status        0
Occupation           1623
Relationship            0
Race                    0
Sex                     0
Hour - per - week       0
Native - Country      580
Income                  0
dtype: int64
```

2) Data Visualization:

Exploring the relationship between the variables.

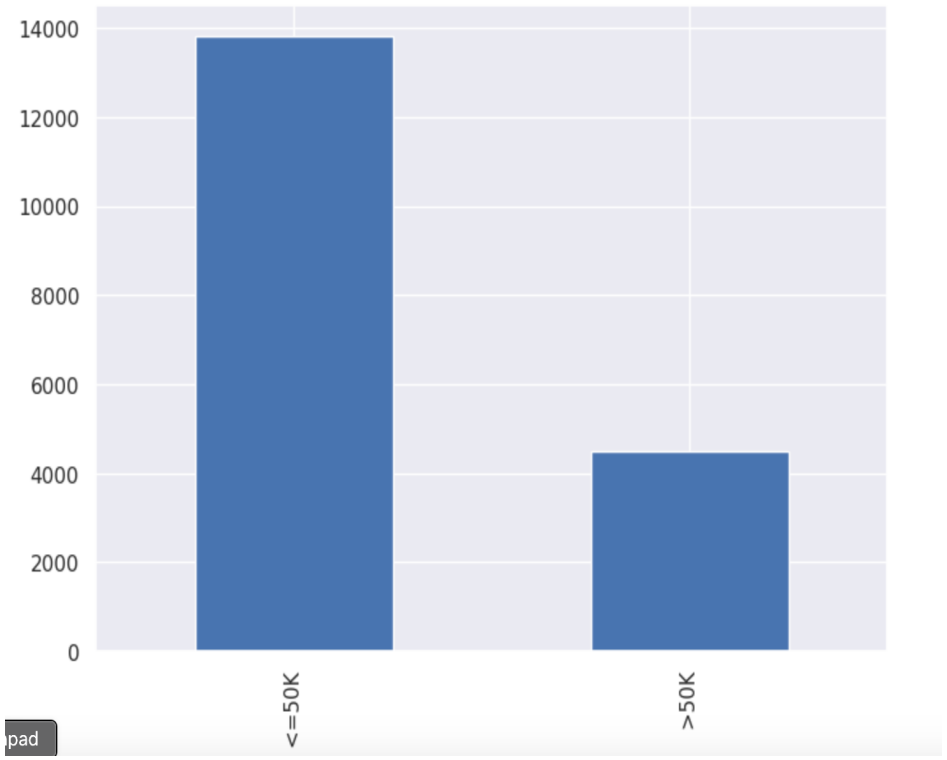


Figure2: Bar Chart displaying count of people with <=50k and >50k

Upon analyzing the dataset's income-based class distribution, we identified an imbalance. To mitigate this, we apply the SMOTE (Synthetic Minority Over-sampling Technique) to address thi issue of underrepresented classes.

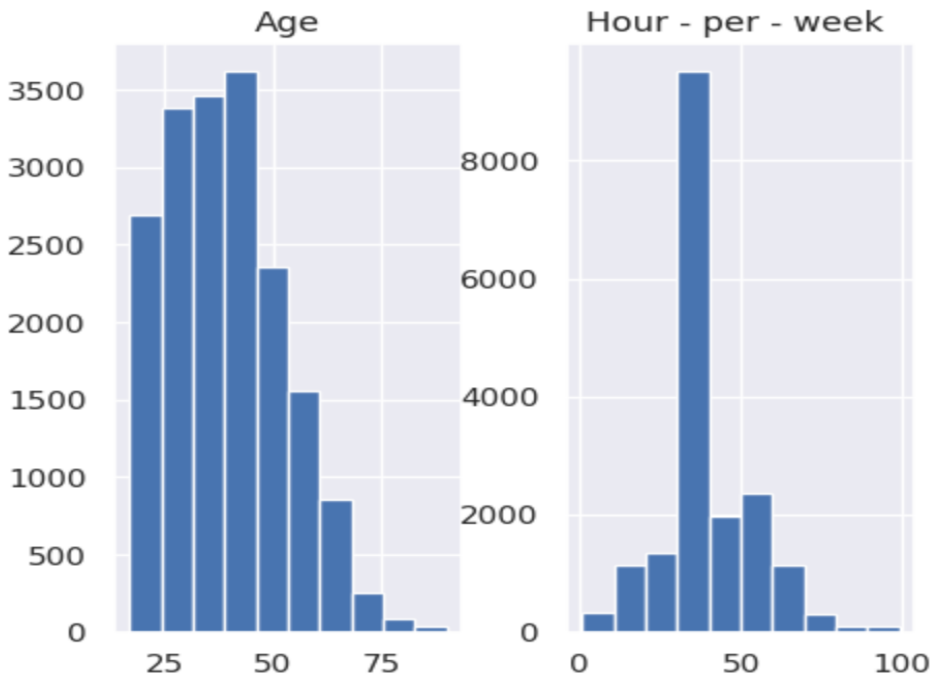


Figure3: Distribution of numerical variables

The data displays values across varying scales, which might not align with the requirements of several machine learning models that prefer uniform scaling across features. Notably, numerous outliers are noticeable within the dataset. To address this, we'll implement the StandardScaler from the sklearn library.

Figure3: Violin plot of ‘work class' w.r.t ‘age’

The width of the violin plot at any given point of time tells us the density of the data points at one particular area. The central white dot represents the mean of the age per workclass category.The shape of the violin overall indicates how skewed the data is. If one side is more extended or thicker than the other, it suggests skewness in the data

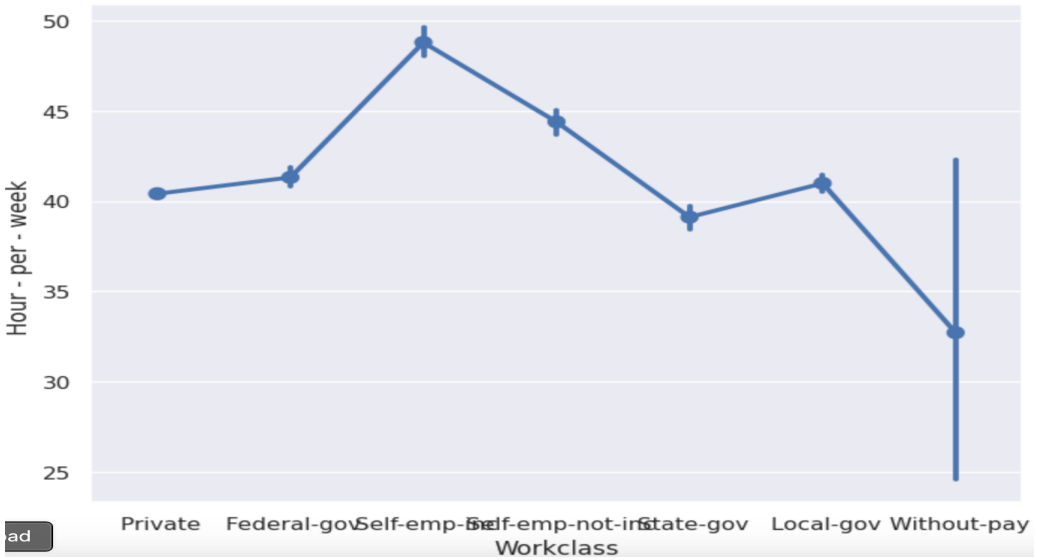


Figure4 : Pointplot of 'Workclass' and 'Hours per week'

Point plots often include markers (points) that represent the central tendency of the data(Here hours per week), such as means or medians. These markers can give you an idea of the average or typical value. Point plots are useful for comparing the central tendency of different groups or categories. If you have multiple groups, the point plot can help you assess whether there are significant differences or trends between them

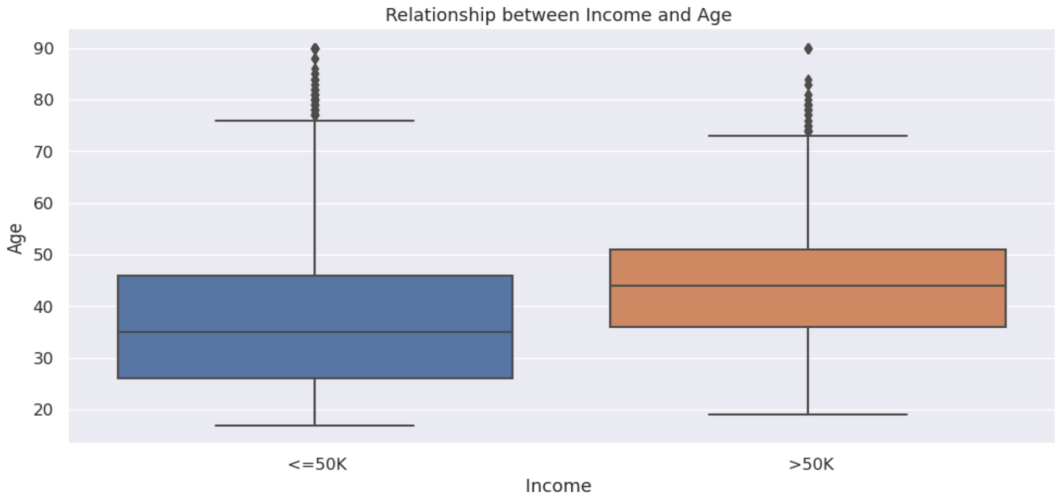


Figure4 : Boxplot of ‘Income’ and 'Age'

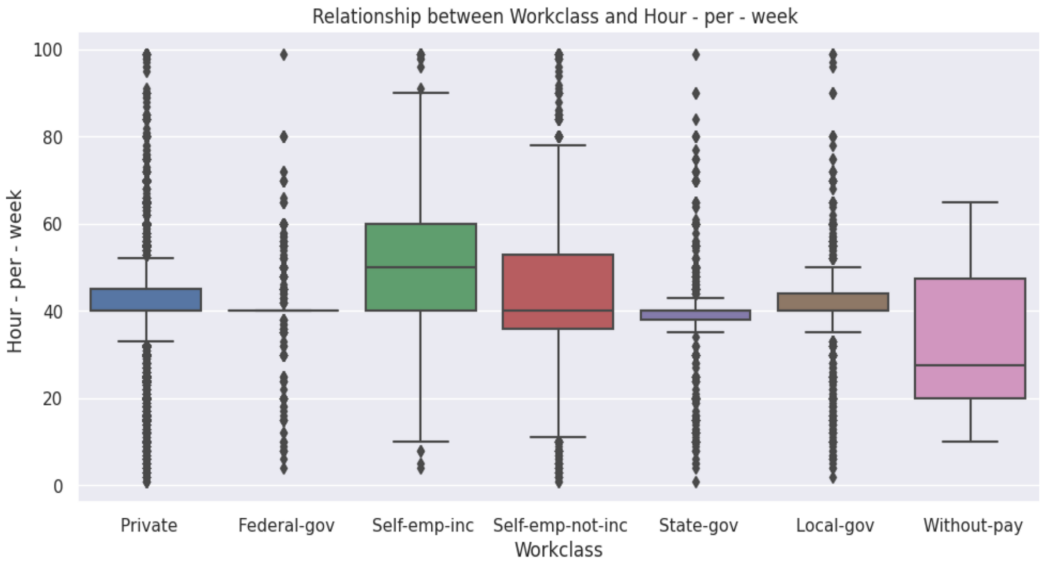


Figure4 : Boxplot of ‘Workclass’ and 'Hours - per - week'

The central line in all box plots depicts the median of Hour - per - week. The box itself represents the interquartile range of Hour - per - week ,which is the range between the first quartile. (Q1) and the third quartile (Q3). It contains the middle 50% of the data and provides a measure of the spread. Individual data points that fall beyond the whiskers are often considered outliers. They can be identified visually on the plot, helping to identify potential anomalies or extreme values in the dataset

3) Feature Selection:

Following columns are dropped because:

- **Fnlwgt** : The range of values in this column was 0 to 4 lakhs which is very vast. Also, while applying naive bayes algorithm, many probabilities were being predicted as 0 and “fmlwgt” showed the estimated number of people in the same occupation as our data point, Therefore, it was not necessary for predicting the income of a particular test data point.
- **Education – num** : This column was just a label encoded version of the ‘Education’ column which did not impact income prediction and was unnecessary repetition of data.
- **Capital – gain** : 95% of the data in this column was either missing or was ‘0’ which would hamper the models efficiency, so we decided to move forward with this column deletion.
- **Capital - loss** : Similar to the column Capital -gain, this column also had maximum values as missing or either ‘0’, which were not required to predict the income class.

4) Data Preprocessing:

- **Encoding** : Categorical features within the dataset, such as 'Education,' 'Marital - status,' 'Relationship,' 'Race,' 'Sex,' 'Native - Country,' 'Workclass,' and 'Occupation' underwent label encoding using the LabelEncoder from the scikit-learn library.

```
# Apply label encoding
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train_encoded)
y_valid_encoded = label_encoder.fit_transform(y_valid_encoded)
y_test_encoded = label_encoder.fit_transform(y_test_encoded)
for feature in ordinal_features:
    X_train_encoded[feature] = label_encoder.fit_transform(X_train_encoded[feature])
    X_valid_encoded[feature] = label_encoder.fit_transform(X_valid_encoded[feature])
    X_test_encoded[feature] = label_encoder.fit_transform(X_test_encoded[feature])
```

This process converts categorical data into numerical form for compatibility with machine learning algorithms. Additionally, the target variable ('Income') is similarly encoded. The encoded features and target variables are copied into separate training, validation, and test sets ('X_train_encoded,' 'X_valid_encoded,' 'X_test_encoded,' 'y_train_encoded,' 'y_valid_encoded,' 'y_test_encoded'), ensuring consistency and independence in subsequent analyses.

- **Scaling** : Following the encoding process, standardization using the StandardScaler from scikit-learn is performed. Standardization helps in scaling numerical features to a standard normal distribution, thereby preventing any particular feature from dominating the learning process due to its larger scale.

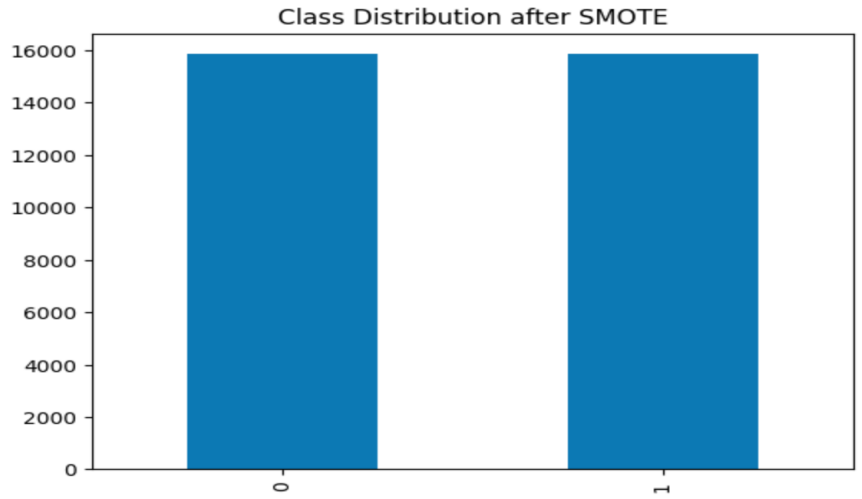
```
scaler = StandardScaler()

X_train_encoded = pd.DataFrame(scaler.fit_transform(X_train_encoded), columns = X_train_encoded.columns)
X_valid_encoded = pd.DataFrame(scaler.fit_transform(X_valid_encoded), columns = X_valid_encoded.columns)
X_test_encoded = pd.DataFrame(scaler.fit_transform(X_test_encoded), columns = X_test_encoded.columns)
```

The training, validation, and test datasets ('X_train_encoded,' 'X_valid_encoded,' 'X_test_encoded') undergo standardization separately, ensuring that each dataset maintains its own mean and variance characteristics. This step aids in improving the performance and convergence speed of various machine learning algorithms by ensuring all features are on a similar scale.

- **Handling Imbalanced Dataset** : The next phase is addressessing the class imbalance issue within the target variable ('Income') using the Synthetic Minority Over-sampling Technique (SMOTE).

The SMOTE oversampled the minority class instances by creating synthetic samples, thus achieving a more balanced distribution between the classes. The 'X_train_encoded' and 'y_train_encoded' datasets are resampled using SMOTE to create balanced training data ('X_resampled' and 'y_resampled'). Subsequently, a bar plot is generated to visualize the post-SMOTE class distribution, showcasing the balanced representation of income classes, thereby improving the training process for machine learning models.



Model Exploration and Selection

This project aims to implement Logistic Regression, Naïve Bayes, Neural Network, and KNN models using a dataset where 'Income' serves as the target variable. Evaluation of these models relies on accuracy as the performance metric, where higher accuracy signifies better model performance. Data splitting involves an initial 30% allocation to the test set and 70% to the training dataset. Within the test set, a further breakdown into an 15% portion for test and 15% for validation is conducted, facilitating robust model training while allowing for meticulous performance assessment

Logistic Regression:

Our base model is chosen as Logistic Regression. Logistic regression is a supervised machine learning algorithm used for binary classification tasks. It predicts the probability of an event occurring based on a set of independent variables. It works by estimating the parameters of a logistic function, which maps the independent variables to the probability of the event happening.

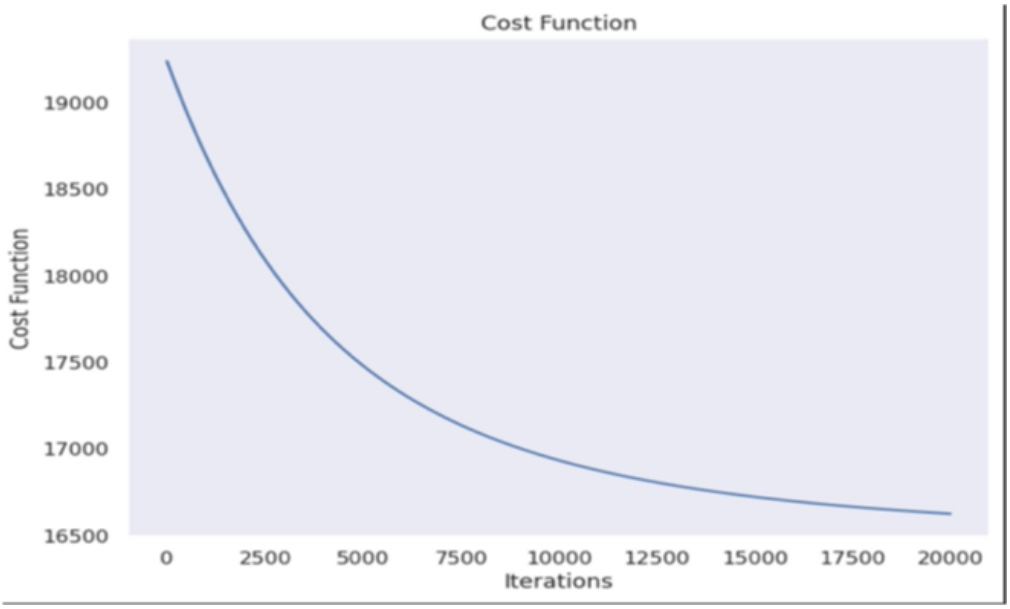
Reason for selection:
It is used when the data is linearly separable, independent, no outliers and the outcome is binary. It provides the binary output using the sigmoid function

Performance without Regularization:

LearningRate	Accuracy_Training_Data	Recall_Training_Data	Accuracy_Test_Data	Recall_Test_Data
0.00000001	0.6943223575185532	0.8258520066287196	0.6440591534931157	0.8348082595870207
0.0000001	0.694754665321709	0.8257079040276677	0.6455889852116268	0.8298918387413963
0.0001	0.694754665321709	0.8259240579292456	0.6463539010708822	0.8298918387413963

Despite variations in the learning rate, there are no substantial changes in the performance on both the training and test datasets.
This stability may indicate that the model is robust to small variations in the learning rate

COST FUNCTION VS ITERATIONS



Performance with L2 Regularization:

LearningRate	Regularization_Param	Accuracy_Train_Data	Recall_Train_Data	Accuracy_Test_Data	Recall_Test_Data
0.00000001	0.01	0.7013473593198357	0.7998414871388428	0.6695563488016318	0.8112094395280236

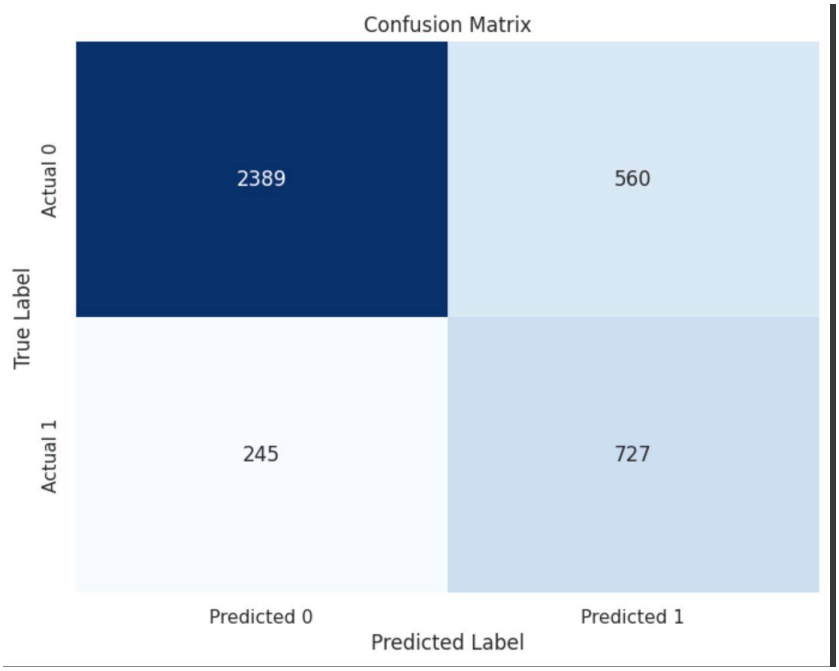
Using L2 regularization in logistic regression has demonstrated an improvement in both accuracy. The regularization term effectively addresses overfitting, leading to a more robust and generalizable model.

Naive Bayes:

It is a classification technique based on Bayes' Theorem with an independence assumption among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

The Naïve Bayes classifier is a popular supervised machine learning algorithm used for classification tasks such as text classification. It belongs to the family of generative learning algorithms, which means that it models the distribution of inputs for a given class or category. This approach is based on the assumption that the features of the input data are conditionally independent given the class, allowing the algorithm to make predictions quickly and accurately.

Performance:




```
Accuracy: 0.7946952308084673
Precision: 0.5648795648795649
Recall: 0.7479423868312757
F1-Score: 0.6436476316954405
```

These metrics suggest that the Naive Bayes model is performing reasonably well, achieving a good balance between precision and recall for the binary classification task

Neural Network:

A neural network classifier is a type of machine learning model that belongs to the broader category of neural networks. Neural networks are computational models inspired by the structure and function of the human brain, and they consist of interconnected nodes (neurons) organized into layers. A neural network classifier, in particular, is designed for classification tasks, where the goal is to assign input data points to specific categories or classes.

Our neural network architecture is designed to effectively capture the complex relationships within the data. The model structure encompasses an input layer with 64 neurons, followed by two hidden layers of 32 and 16 neurons, respectively. The output layer, crucial for predicting the binary outcome, consists of a single neuron utilizing the sigmoid activation function.

Neural Network Configuration:

1. **Input Layer (Dense):**
    - Neurons: 64
    - Activation Function: ReLU
    - Purpose: Gathers input features and initiates the flow of information.
2. **Dropout Layer:**
    - Dropout Rate: 0.5
    - Purpose: Mitigates overfitting by randomly dropping out connections during training.
3. **Hidden Layers (Dense):**
    - Neurons: 32 (first hidden layer), 16 (second hidden layer)
    - Activation Function: ReLU
    - Purpose: Introduces non-linearity to the model, enabling it to capture intricate patterns in the data.
4. **Output Layer (Dense):**
    - Neurons: 1
    - Activation Function: Sigmoid
    - Purpose: Produces a binary output, representing the predicted class probabilities.

Model Compilation:

- **Optimizer:** Adam optimizer with a learning rate of 0.001.
- **Loss Function:** Binary cross-entropy, suitable for binary classification tasks.
- **Metrics:** Accuracy is chosen as the evaluation metric during training.

Model Training:

- Trained for 20 epochs with a batch size of 64.
- Validation data (X_valid_encoded, y_valid_encoded) used to monitor model performance during training.

Evaluation Metrics:

- Model performance evaluated on the test set using various metrics:
    - Accuracy, Precision, Recall, F1 Score, and ROC AUC Score.

Confusion Matrix:

- Confusion matrix generated to provide a detailed breakdown of model predictions.

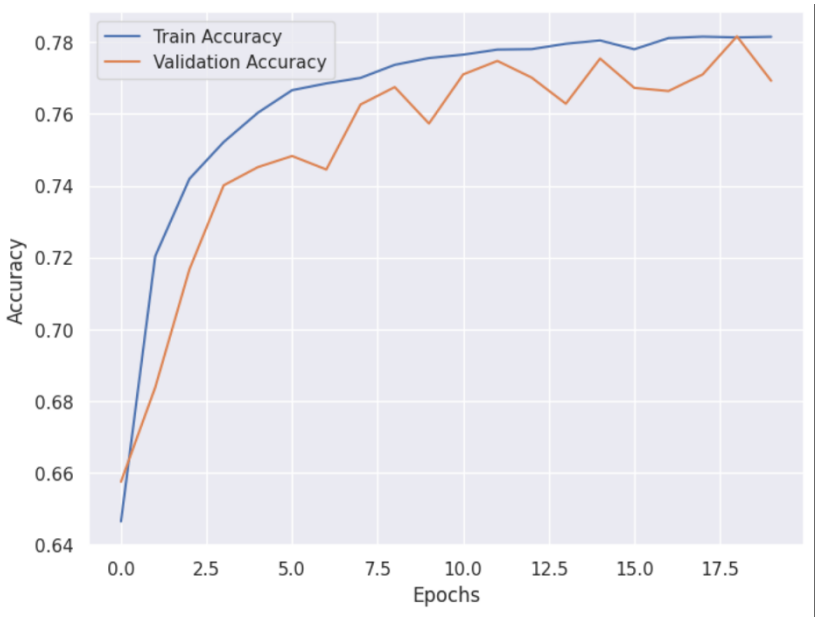
Training History Plot:

- Visualization of the model's accuracy over epochs, showcasing the training and validation accuracy trends.

Performance:

```
Accuracy: 0.7508841732979664
Precision: 0.49783783783783786
Recall: 0.8230563002680965
F1 Score: 0.6204109127652409
ROC AUC Score: 0.8470632080510315
Confusion Matrix:
[[2476  929]
 [ 198  921]]
```

Comparison of Training and Validation Accuracy by number of Epochs:



The model demonstrates good discrimination, overall correctness, and proficiency in identifying positive instances, but with room for improvement in positive prediction accuracy

Results

Overall Comparison of models with accuracy metrics

MODEL	ACCURACY	RECALL
Logistic Regression	0.6440591534931157	0.8348082595870207
Logistic Regression with L2 Regularization	0.6695563488016318	0.8112094395280236
Naïve Bayes	0.7946952308084673	0.7479423868312757
Neural Networks	0.7656807751147374	0.8190757128810227

Naïve Bayes outperforms other models with the highest accuracy (79%), while Neural Networks exhibit a strong balance between overall accuracy (77%) and positive instance recall (82%).