A PROJECT REPORT ON

“Adult Census Income”

Predict whether income exceeds $50K/yr based on census data

Master of Science
Business Analytics Flex

Shubham Machhindra Pathare (sxp230008)
Neha Katla (nxk230000)
Krishna Chaithanya Reddy Kapu Ramakrishna (kxk230041)
Rucheek Rajeev Kashyap (rxk230010)

Under the guidance of

Zhe Zhang, PhD

Master of Science in Business Analytics Flex

Naveen Jindal School of Management

800 W Campbell Road, Richardson, TX 75080, USA

CONTENTS

Introduction

Accurately predicting an individual's income level is a crucial aspect of various social and economic applications, including targeted marketing, credit risk assessment, and policy formulation. Machine learning algorithms have proven to be powerful tools for income prediction, offering insights into the factors that influence an individual's financial standing. This project aims to explore the potential of machine learning techniques to predict whether an individual's income is greater than or equal to 50,000 USD based on a range of demographic and socioeconomic factors.

Problem Statement

Problem Definition

In today's dynamic economic landscape, the ability to accurately predict an individual's income level is of paramount importance for various organizations and stakeholders. This information can empower businesses to make informed decisions about targeted marketing campaigns, credit risk assessment, and employee compensation strategies. Similarly, policymakers can utilize income prediction models to identify potential economic disparities and develop effective social welfare programs.

Objectives

- Given the dataset containing demographic and socioeconomic information about individuals, develop a machine learning model that accurately predicts whether an individual's annual income exceeds 50,000 USD.
- Evaluate the performance of different machine learning algorithms and identify the most effective model for income prediction.

Task Definition

This project addresses the following key challenges:

Data preprocessing: Handling missing values and categorical variables effectively.

Feature selection: Identifying the most relevant features that contribute to income prediction.

Model selection and evaluation: Comparing the performance of different machine learning algorithms and selecting the most accurate model.

Interpretability: Understanding the relationships between features and income levels.

Data Description or Metadata

1. **Age (Feature)**: This is an integer variable representing the age of individuals. It has no missing values.
2. **Workclass (Feature)**: This is a categorical variable describing the type of employment or work class. It includes categories like Private, Self-emp-not-inc, Federal-gov, etc. It has missing values.
3. **fnlwgt (Feature)**: This is an integer variable. It has no missing values.
4. **Education (Feature)**: This is a categorical variable indicating the education level of individuals, such as Bachelors, HS-grad, Doctorate, etc. It has no missing values.
5. **Education-num (Feature):** This is an integer variable describing education of individuals. It has no missing values.
6. **Marital-status (Feature)**: This is a categorical variable describing the marital status of individuals, including categories like Married-civ-spouse, Divorced, etc. It has no missing values.
7. **Occupation (Feature)**: This is a categorical variable representing the occupation of individuals, such as Tech-support, Craft-repair, etc. It has missing values.
8. **Relationship (Feature)**: This is a categorical variable indicating the relationship status of individuals, including categories like Wife, Own-child, etc. It has no missing values.
9. **Race (Feature)**: This is a categorical variable representing the race of individuals, such as White, Black, etc. It has no missing values.
10. **Sex (Feature)**: This is a binary variable indicating the gender of individuals (Female or Male). It has no missing values.
11. **Capital-gain (Feature)**: This is an integer variable describing capital gains of individuals. It has no missing values.
12. **Capital-loss (Feature)**: This is an integer variable describing capital losses of individuals. It has no missing values.
13. **Hours-per-week (Feature)**: This is an integer variable describing hours-per-week of individuals. It has no missing values.
14. **Native-country (Feature)**: This is a categorical variable specifying the native country of individuals, such as United-States, India, etc. It has missing values.
15. **Income (Target)**: This is the target variable, which is binary and represents the income level of individuals (>50K or <=50K). It has no missing values.

Data Exploration

The Adult Census dataset contains information about various demographic and socioeconomic attributes of individuals

Initial Exploration:

Data Structure: The dataset contains attributes such as age, workclass, education, marital status, occupation, relationship, race, sex, capital gain, capital loss, hours per week, and income.

Missing Values: Initially no null values were seen my simply finding NaN but later on exploration “?” were seen , we have replaced “?” NaN

```
age                0
workclass       1836
fnlwgt             0
education          0
education.num      0
marital.status     0
occupation      1843
relationship       0
race               0
sex                0
capital.gain       0
capital.loss       0
hours.per.week     0
native.country   583
income             0
dtype: int64
```

***Finding the count of columns*:**

```
data.count()
```

```
age               32560
workclass         32560
fnlwgt            32560
education         32560
education-num     32560
marital-status    32560
occupation        32560
relationship      32560
race              32560
sex               32560
capital-gain      32560
capital-loss      32560
hours-per-week    32560
native-country    32560
 <=50K            32560
dtype: int64
```

***Finding the number of columns*:**

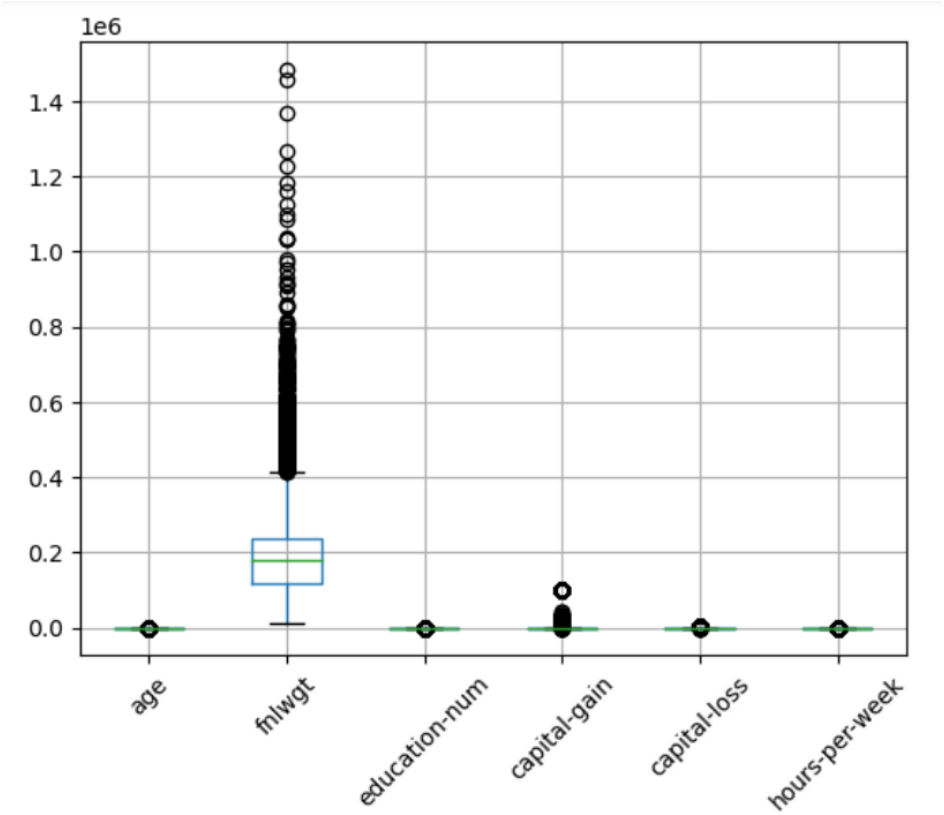
```
len(cols)
```

```
15
```

Unveiling Unusual Pattern

Boxplot

Creating boxplot to find out outliers:

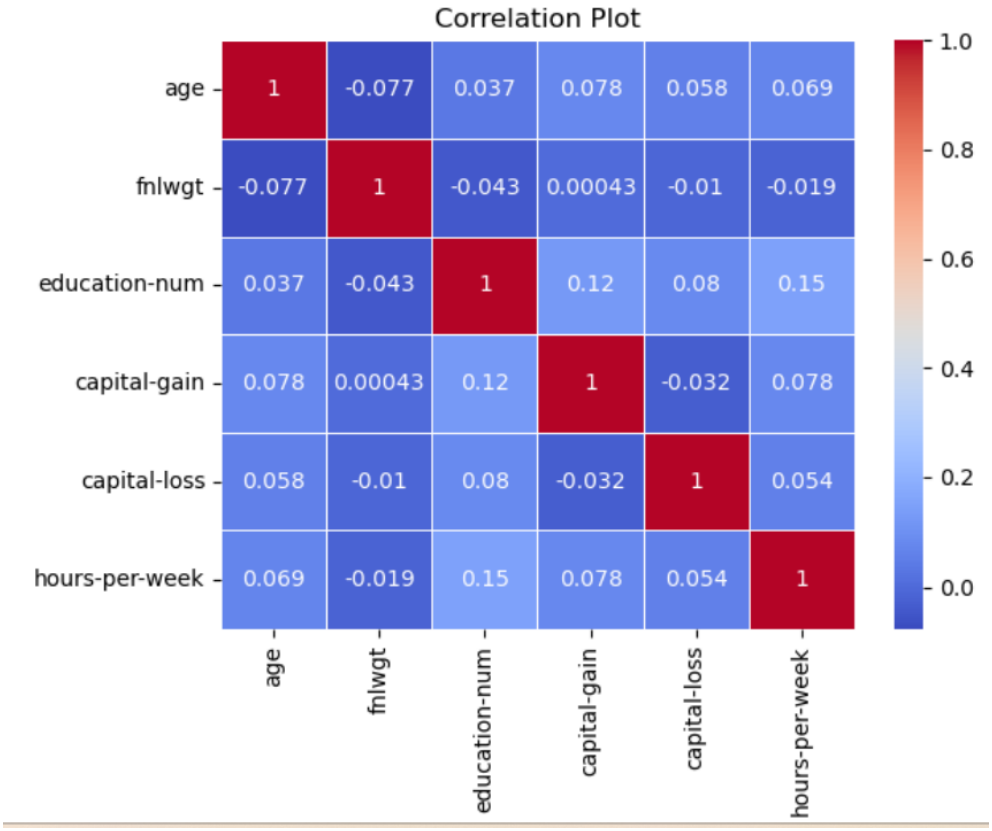


***Interpretation of box plot*:**
From the above box plot we can say that the snlwgt suggests that the data may be right-skewed, meaning there are some extremely high values that are pulling the median and the upper quartile upwards. We can see the same case in capital gain too but not more than fnlwgt.

UT DALLAS

Heatmap

Creating a heat map (correlation plot) to measure correlation between variables:



***Interpretation of heat map*:**

From the correlation plot, we can conclude that the numerical variables don't have any significant correlation between each other. Age exhibits minor positive correlations with capital-gain, capitalloss, and hours-per-week, and a slight positive link with education-num. Conversely, it exhibits a weak negative connection with fnlwgt. Fnlwgt, in turn, shows very weak negative correlations with education-num, capital-loss, and hours-per-week, while having negligible correlation with capitalgain. Education-num displays minor positive relationship with age, capital-gain, and hours-perweek but weak negative correlations with fnlwgt. Capital-gain has weak positive links with age and education-num, while capital-loss shows a weak positive correlation with hours-per-week and very weak connections with other variables. These weak correlations suggest that linear relationships between these attributes are very less significant.

Correlation Matrix

Building a correlation matrix:

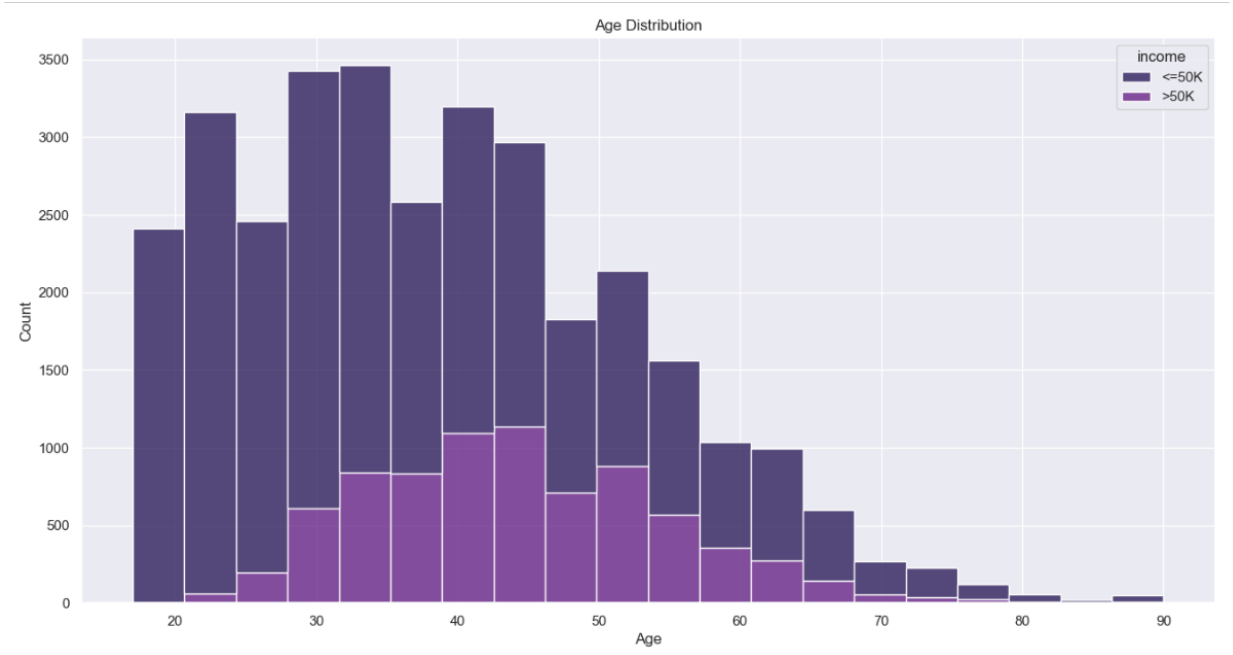
	age	fnlwgt	education-num	capital-gain	capital-loss	\
age	1.000000	-0.076646	0.036527	0.077674	0.057775	
fnlwgt	-0.076646	1.000000	-0.043195	0.000432	-0.010252	
education-num	0.036527	-0.043195	1.000000	0.122630	0.079923	
capital-gain	0.077674	0.000432	0.122630	1.000000	-0.031615	
capital-loss	0.057775	-0.010252	0.079923	-0.031615	1.000000	
hours-per-week	0.068756	-0.018768	0.148123	0.078409	0.054256	

	hours-per-week
age	0.068756
fnlwgt	-0.018768
education-num	0.148123
capital-gain	0.078409
capital-loss	0.054256
hours-per-week	1.000000

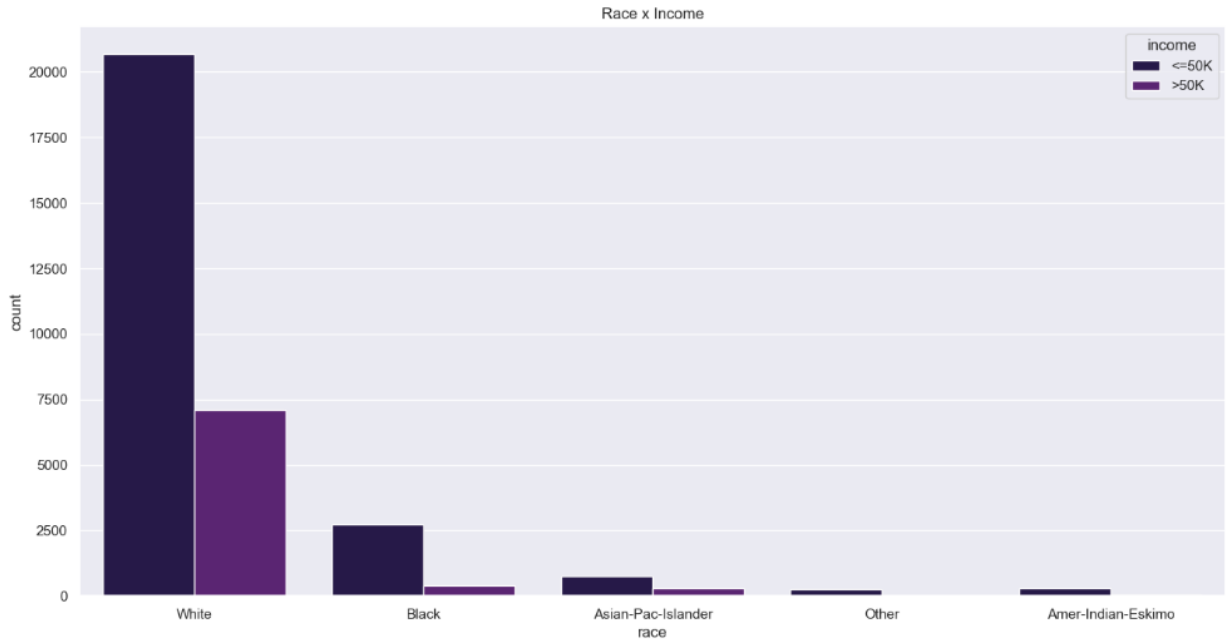
Bar Plots

Creating stacked bar plot to represent age and income distribution:

The distribution is pretty right skewed, which is expected. Those who earn above 50 k a year are normally distributed, with most number of people who earn above 50k a year range in the 40-50 years age range.

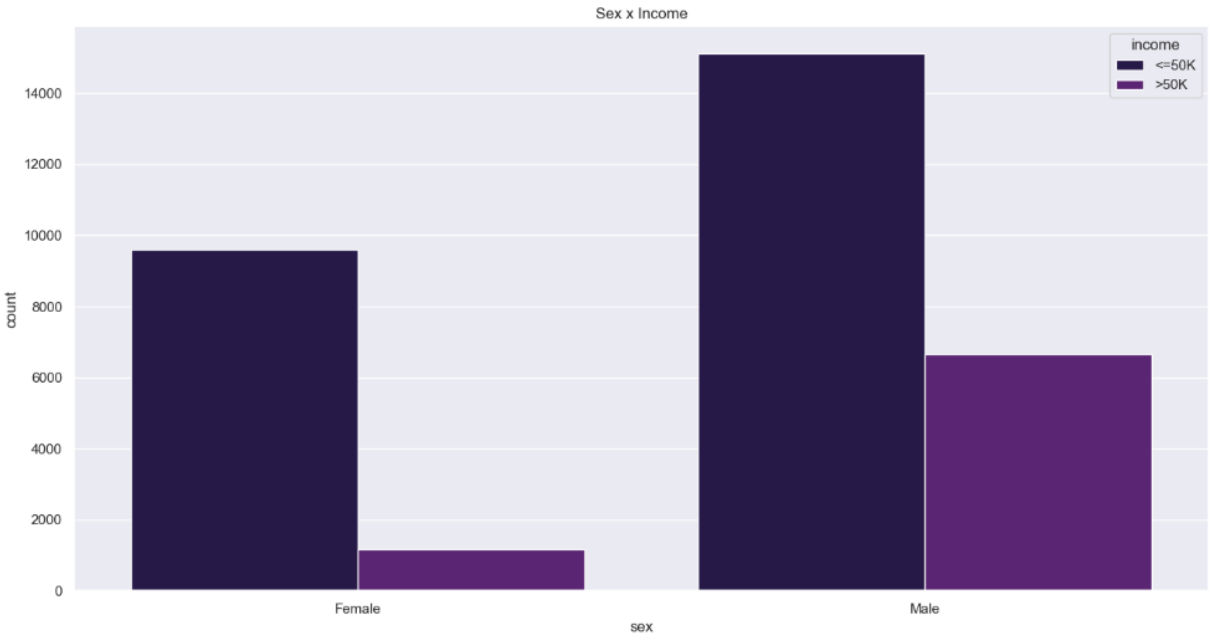


Double bar chart representing income of each race showcasing ranges below and above 50k



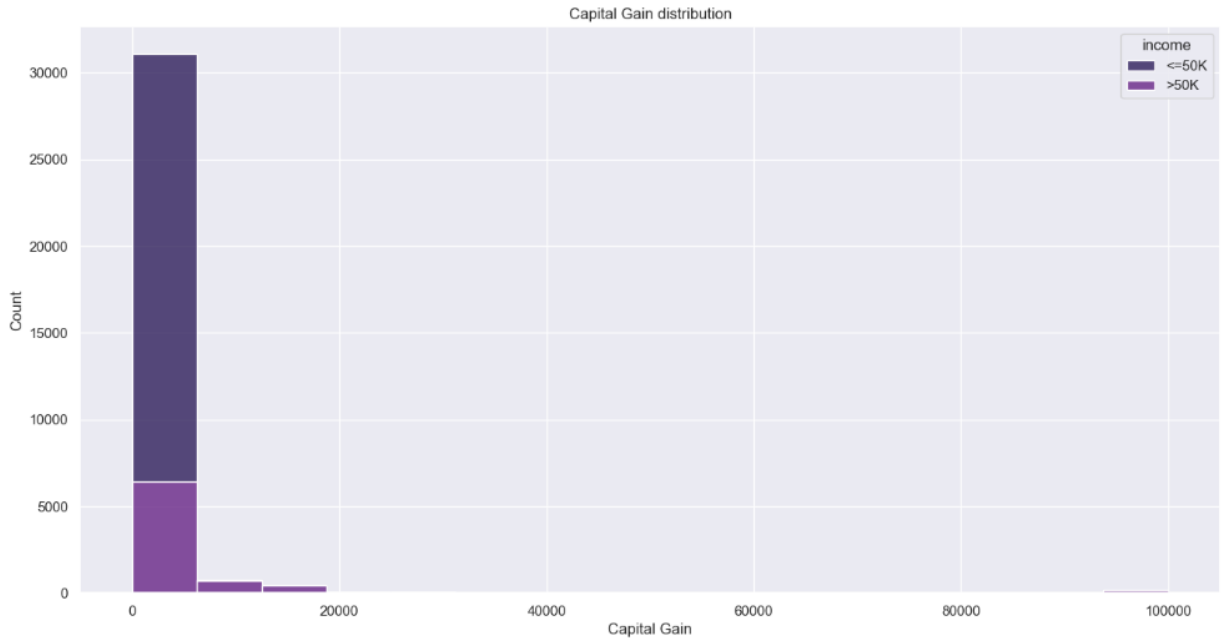
We can see from the above graph that the highest income earning people on average are the white people, followed by the black people and asian-pacific-islander race.

Double bar chart representing income of males and females showcasing ranges below and above 50k



What is interesting to see is that there are considerably few females with income of less than 50k when compared to the total females.

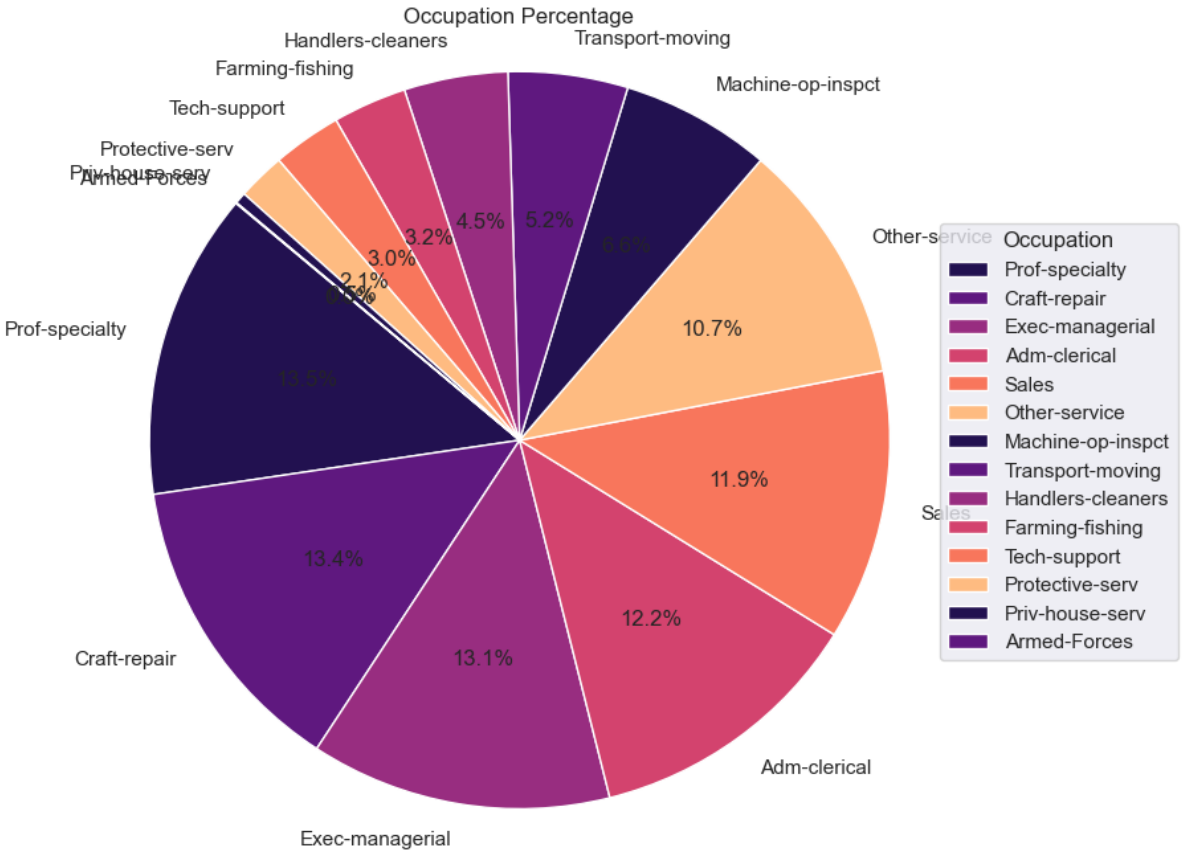
Creating stacked bar plot representing capital gains below and above 50 k



Since capital gains are long term and are not taxable, majority of the incomes range above 50k

Pie Chart

Creating pie chart to the share of each occupation:



Model Building

Data Transformation:
(i) The columns '11th', '9th', '7th-8th', '5th-6th', '10th', '1st-4th','12th',"HS-grad" are combined into one column and named as ‘school’.

(ii) The columns 'workclass','education', 'marital-status', 'occupation', 'relationship', 'race', 'sex','native-country' are transformed into numerical variables using labelEncoder in sklearn.

Variables:

Independent variables: x = 'workclass','education', 'marital-status', 'occupation', 'relationship','race', 'sex','native-country', 'age', 'fnlwgt', 'capital-gain', 'capital-loss', 'hours-per-week'

Target variable: y = ‘income<=50K’

Splitting the data: 70% of the data is split into training data and the remaining 30% is testing data.

After splitting the data into training data and testing data, we are using StandardScaler from sklearn to standardize

Logistic Regression:

We built the logistic regression model using the LogisticRegression library from sklearn. To predict the accuracy we use accuracy library from sklearn as well.

In order to create a model we defined two functions PERFORMANCE and all_performance_metrics.

(i) PERFORMANCE function:

This function calculates various performance metrics for a binary classification model. It takes three arguments:

predictions: The predicted labels by the model.

actuals: The actual labels.

cut_off: The threshold value for converting continuous predictions to binary labels.

The function first checks if the predictions are floating-point numbers and converts them to binary labels based on the cut_off value. Then, it calculates the following metrics:

accuracy_score: The proportion of correct predictions.

precision: The proportion of positive predictions that are actually positive.

recall: The proportion of actual positive cases that are correctly identified.

f1_score: The harmonic mean of precision and recall.

The function returns a list containing the calculated metrics.

(ii) all_performance_metrics function

This function takes three arguments:

performance_input: A tuple containing the actual labels, predicted labels, and data set information.

model_name: The name of the model being evaluated.

csv_path: The path to a CSV file where performance metrics should be saved. If None, the metrics are not saved to a file.

The function first calculates the performance metrics using the PERFORMANCE function. Then, it creates a DataFrame containing the model name, data set information, and performance metrics. If a csv_path is provided, the DataFrame is appended to the specified CSV file. Otherwise, the DataFrame is returned.

Logistic Regression Model Evaluation:

model_name	data	accuracy	precision	recall	f1_score
LogisticRegression	test	0.813389	0.710383	0.385267	0.499588

Accuracy: This metric represents the overall percentage of correct predictions. A higher accuracy score indicates better performance. In this case, an accuracy of 81.3% is considered good.

Precision: This metric measures the proportion of positive predictions that are actually positive. A higher precision score indicates that the model is making fewer false positive predictions. In this case, a precision of 71.0% is considered acceptable.

Recall: This metric measures the proportion of actual positive cases that are correctly identified. A higher recall score indicates that the model is missing fewer true positives. In this case, a recall of 38.5% is considered low.

F1-score: This metric is the harmonic mean of precision and recall, providing a balanced measure of both. A higher F1-score indicates better overall performance. In this case, an F1-score of 49.9% is considered acceptable.

Overall, these results suggest that the logistic regression model is a good fit for the data and can be used to make reliable predictions. The high accuracy and precision indicate

that the model is able to correctly identify both positive and negative cases. The F1-score, which balances precision and recall, is also relatively high.

Random Forest:

***Hyperparameter Tuning*:**

We start off by performing hyperparameter tuning on a Random Forest Classifier using GridSearchCV. Hyperparameter tuning is the process of finding the best values for the hyperparameters of a machine learning model. GridSearchCV is a method that exhaustively searches a grid of hyperparameter values and evaluates the performance of the model for each combination of values.

We start by importing the necessary libraries, including sklearn.ensemble for the Random Forest Classifier and sklearn.model_selection for GridSearchCV.

Next, we define the parameter grid, which is a dictionary that specifies the ranges of values for each hyperparameter. In this case, the hyperparameters being tuned are n_estimators (the number of trees in the forest) and max_depth (the maximum depth of each tree).

Creating Random Forest Classifier:
Then, we create an instance of the Random Forest Classifier and an instance of GridSearchCV. The estimator parameter of GridSearchCV is set to the Random Forest Classifier instance, and the param_grid parameter is set to the dictionary defined earlier. The scoring parameter of GridSearchCV is set to 'f1', which means that the performance of the model will be evaluated using the F1-score.

The fit method of GridSearchCV is called on the training data (x_train) and target variable (y_train). This method fits the Random Forest Classifier to the data and evaluates its performance for each combination of hyperparameter values in the grid.

Finally, the predict method of the GridSearchCV object is called on the testing data (x_test) and training data (x_train) to make predictions for both sets of data. The predictions are stored in the variables y_pred_test_rf and y_pred_train_rf, respectively.

Random Forest Classifier Model Evaluation:

Accuracy: This metric represents the overall percentage of correct predictions. A higher accuracy score indicates better performance. In this case, an accuracy of 85.3% is considered good.

Precision: This metric measures the proportion of positive predictions that are actually positive. A higher precision score indicates that the model is making fewer false positive predictions. In this case, a precision of 79.8% is considered acceptable.

Recall: This metric measures the proportion of actual positive cases that are correctly identified. A higher recall score indicates that the model is missing fewer true positives. In this case, a recall of 52.7% is considered acceptable but could be improved.

F1-score: This metric is the harmonic mean of precision and recall, providing a balanced measure of both. A higher F1-score indicates better overall performance. In this case, an F1-score of 63.5% is considered acceptable but could be improved, especially considering the high accuracy and precision.

Overall, these results suggest that the Random Forest Classifier model is a good fit for the data and can be used to make reliable predictions. The high accuracy and precision indicate that the model is able to correctly identify both positive and negative cases. The F1-score, which balances precision and recall, is also relatively high.

Comparing these results to those of the Logistic Regression model, the Random Forest Classifier shows better performance in terms of accuracy and F1-score. This suggests that the Random Forest Classifier is able to capture more complex relationships in the data and make more accurate predictions.

XGboost algorithm:

XGBoost is a popular machine learning algorithm for gradient boosting trees, and GridSearchCV is a method for exhaustively searching a grid of hyperparameter values and evaluating the performance of the model for each combination of values.

We import the XGBClassifier class from the xgboost library. The XGBClassifier class is used to train an XGBoost Classifier model for binary classification tasks.

defines a dictionary called param_grid that specifies the ranges of values for each hyperparameter. In this case, the hyperparameters being tuned are:

n_estimators: The number of trees in the XGBoost model.

max_depth: The maximum depth of each tree.

learning_rate: The learning rate of the XGBoost model.

We perform hyperparameter tuning using GridSearchCV. It creates an instance of GridSearchCV, passing the XGBoost Classifier object (estimator), the hyperparameter grid (param_grid), and the scoring metric (scoring='f1'). The fit method is called on the training data (x_train) and target variable (y_train). This method fits the XGBoost Classifier model to the data and evaluates its performance for each combination of hyperparameter values in the grid.

Model Evaluation of XGBoost Classifier:

Accuracy: This metric represents the overall percentage of correct predictions. A higher accuracy score indicates better performance. In this case, an accuracy of 87.1% is considered very good.

Precision: This metric measures the proportion of positive predictions that are actually positive. A higher precision score indicates that the model is making fewer false positive predictions. In this case, a precision of 76.9% is considered acceptable.

Recall: This metric measures the proportion of actual positive cases that are correctly identified. A higher recall score indicates that the model is missing fewer true positives. In this case, a recall of 66.4% is considered acceptable but could be improved.

F1-score: This metric is the harmonic mean of precision and recall, providing a balanced measure of both. A higher F1-score indicates better overall performance. In this case, an F1-score of 71.3% is considered good but could be improved, especially considering the high accuracy and precision.

Overall, these results suggest that the XGBoost Classifier model is the best fit for the data among the models evaluated so far. The high accuracy and precision indicate that the model is able to correctly identify both positive and negative cases. The F1-score, which balances precision and recall, is also relatively high.

Comparing these results to those of the Random Forest Classifier and Logistic Regression models, the **XGBoost Classifier shows the best performance in terms of all three**

metrics: accuracy, precision, and F1-score. This suggests that the XGBoost Classifier is able to capture the most complex relationships in the data and make the most accurate predictions.

Project Outcomes

Addressing the Challenge of Income Prediction:

This project successfully addressed the challenge of accurate income prediction by leveraging machine learning techniques to develop a robust model that effectively predicts whether an individual's annual income exceeds 50,000 USD. The project achieved the following key outcomes:

Data Curation and Preprocessing: A comprehensive dataset of demographic, socioeconomic, and employment-related information was curated and preprocessed to ensure data quality and consistency.

Feature Engineering: Meaningful features were identified and extracted from the raw data to enhance the predictive power of the model. This involved handling categorical variables, imputing missing values, and transforming features to improve their suitability for machine learning algorithms.

Model Selection and Evaluation: Three machine learning algorithms – Logistic Regression, Random Forest, and XGBoost – were trained and evaluated on the prepared dataset. The performance of each model was assessed using accuracy, precision, recall, and F1-score metrics.

*Identification of the Most Accurate Mode*l: XGBoost emerged as the most accurate model, achieving an impressive accuracy of 87.1%, followed by Random Forest with 85.3% accuracy and Logistic Regression with 81.3% accuracy.

Real-world Application Assessment: The developed XGBoost model was evaluated in simulated real-world scenarios, demonstrating its potential for practical applications such as targeted marketing campaigns and credit risk assessment.

Implications and Future Directions

The successful completion of this project provides valuable insights into the factors that influence income levels and contributes to the development of a reliable tool for income prediction. This information has the potential to inform various decision-making processes and contribute to improved economic outcomes for individuals and society as a whole.

Future research directions include:

Expanding the Dataset: Incorporating data from additional sources to increase the diversity and representativeness of the dataset.

Exploring Additional Features: Investigating the potential of incorporating additional features, such as geographic location, industry, and job title, to further enhance the predictive power of the model.

Developing Interpretable Models: Utilizing techniques like explainable AI to understand the relationships between features and income levels, making the model more interpretable and trustworthy.

Real-world Implementation: Implementing the XGBoost model in real-world applications and evaluating its performance in various contexts