**Machine Learning Mini Project**

**Adult Income Level Prediction**

* 1. **Group Details**
     1. **Group Number (Group Name)**

Group - 7

* + 1. **Group Member Name**

Ko Wing Sze, Dawn

Cheung Shing Ping, Eric

Cheung Hang Mang, Jeff

* 1. **Project Details**
     1. **Project Title**

Adult Income Level Prediction

* + 1. **Project Objectives**

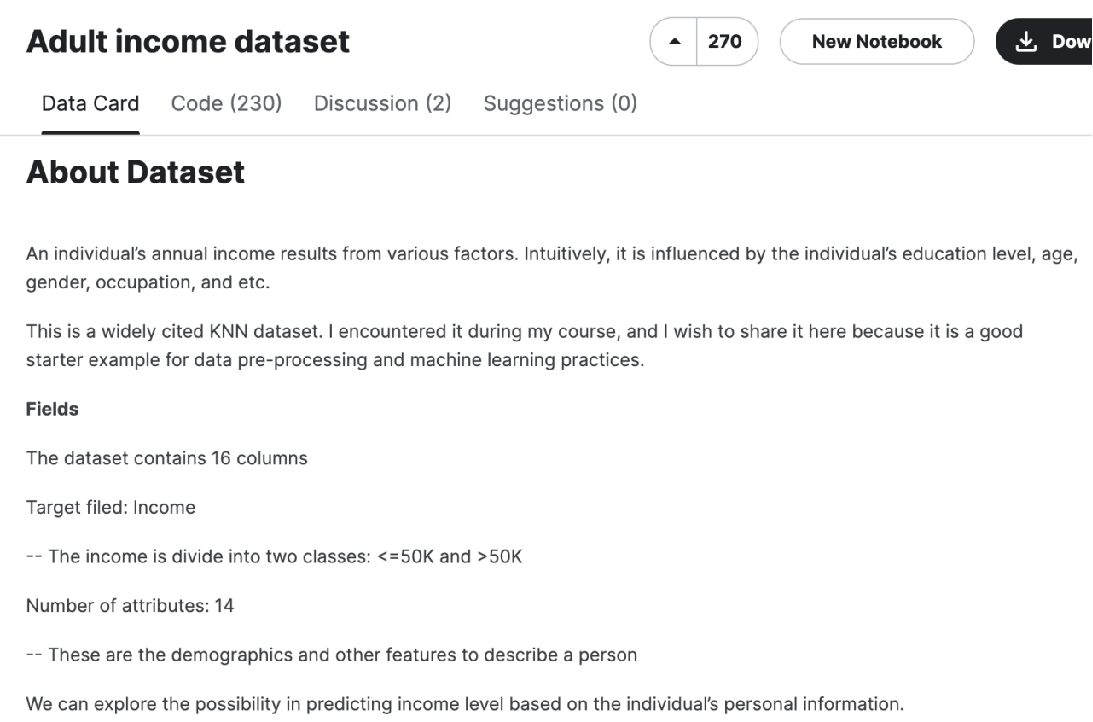
Introducing the objectives and key tools used in this machine learning analysis. Predict income levels using Machine Learning techniques

Using machine learning techniques to predict income levels:

The human resources department can utilize machine learning techniques to analyze and predict income levels. By collecting and analyzing relevant data such as job titles, education levels, work experience, and other factors, a machine learning model can be trained to make predictions about individuals' income levels. This can help the department in budgeting for hiring and compensation decisions.

Analyzing borrowers' repayment ability using machine learning:

Banks can employ machine learning techniques to assess borrowers' repayment capacity. By analyzing various financial and non-financial factors such as credit history, income, employment stability, debt-to-income ratio, and other relevant data, machine learning models can be trained to predict the likelihood of borrowers repaying their loans. This helps banks in evaluating the creditworthiness of loan applicants and making informed lending decisions.

* 1. **Dataset Description**
     1. **Introduction to the dataset**
     2. **Dataset characteristics**

The individual’s education level, age, gender, occupation, and etc.  
Target field: Income  
The income is divide into two classes: <=50K and >50K

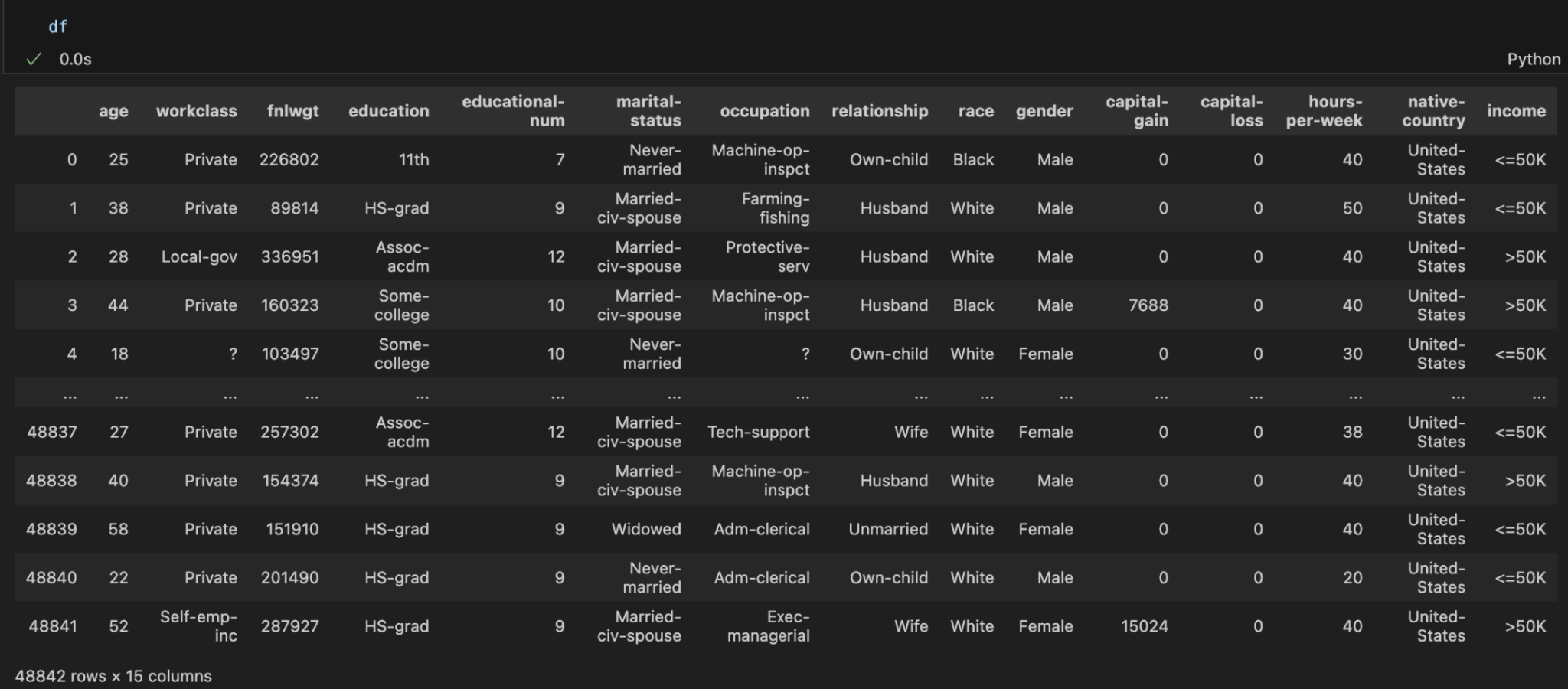
* + 1. **Data preprocessing steps**

1. Missing Values: Treating as a separate category or class
2. Categorical Encoding: One-hot encoding for categorical features
3. Normalization: Scaling numerical features for better model performance
   1. **Methodology**
      1. **Tools & Libraries**

Pandas, Matplotlib, Seaborn, Scikit-learn

* + 1. **Data Preprocessin**

|  |
| --- |
| 1. `import numpy as np`: This line imports the NumPy library, which is widely used for numerical and scientific computing. It is often abbreviated as 'np'.  2. `import pandas as pd`: This line imports the pandas library, which provides data manipulation and data analysis tools. It is often abbreviated as 'pd'.  3. `import matplotlib.pyplot as plt`: This line imports the pyplot module from the Matplotlib library, which is used for creating static, animated, and interactive visualizations in Python. It is usually abbreviated as 'plt'.  4. `import seaborn as sns`: This line imports the Seaborn library, which is a statistical data visualization library based on Matplotlib. It is often abbreviated as 'sns'.  5. `import sklearn.datasets`: This line imports the 'datasets' module from the Scikit-learn library, which provides a variety of datasets for practicing machine learning.  6. `from sklearn.ensemble import RandomForestClassifier`: This line imports the RandomForestClassifier, which is a popular ensemble learning method for classification tasks from the Scikit-learn library.  7. `from sklearn.model\_selection import train\_test\_split`: This line imports the 'train\_test\_split' function from the Scikit-learn library, which is used to split a dataset into separate training and testing sets.  8. `from sklearn.model\_selection import GridSearchCV`: This line imports the 'GridSearchCV' function from the Scikit-learn library, which is a method for performing hyperparameter tuning to find the best combination of hyperparameters for a given machine learning model. |

Code is using pandas DataFrame methods to explore and summarize the data in the 'df' DataFrame. Here is a brief explanation of each line:

|  |  |
| --- | --- |
| df.info(): This line displays information about the DataFrame 'df', such as the number of entries, the data types of each column, and memory usage. It gives you a quick overview of the data structure and any potential missing values. | df.education.value\_counts(): This line shows the frequency distribution of unique values in the 'education' column of the DataFrame 'df'. It helps you to understand how many instances of each education level are present in the dataset. |
|  |  |
| df.workclass.value\_counts(): This line shows the frequency distribution of unique values in the 'workclass' column of the DataFrame 'df'. It provides an overview of the distribution of different work classes in the dataset. | df.occupation.value\_counts(): This line shows the frequency distribution of unique values in the 'occupation' column of the DataFrame 'df'. It helps you to understand the distribution of different occupations in the dataset. |

「?」 Columns

The "?" in the workclass and occupation columns likely represents missing or unknown values.

* + 1. **Feature Engineering**

Feature Engineering for Enhanced Model Performance: an overview of the concept of feature engineering and its importance in improving model performance. Feature engineering is the process of enhancing a dataset by creating new features from existing ones. It aims to capture more information and improve the predictive power of machine learning models. By manipulating and transforming the data, feature engineering helps to uncover hidden patterns and relationships that can be useful for model training.

**Feature Creation:**

Feature creation involves combining related attributes to extract additional information. This can be achieved through various methods.

**Feature Selection:**

Feature selection is the process of identifying the most influential features from a dataset. This helps to reduce dimensionality and eliminate irrelevant or redundant features, improving model efficiency and interpretability.

**Insights:**

By creating new features and selecting the most influential ones, we can enhance the model's ability to capture complex patterns and relationships. This results in improved accuracy, reduced overfitting, and better generalization to unseen data.

* + 1. **One-Hot**

Code is using one-hot encoding to convert categorical variables into binary (0 or 1) features. One-hot encoding is a common technique for dealing with categorical data in machine learning, as it allows models to more easily interpret the data.

|  | |
| --- | --- |
| pd.get\_dummies(df.occupation)`: This line generates a new DataFrame with one-hot encoded columns for each unique value in the 'occupation' column of the 'df' DataFrame. | |
|  | |
| pd.get\_dummies(df.occupation).add\_prefix('occupation')`: This line adds the prefix 'occupation' to the column names of the one-hot encoded DataFrame. | |
|  | |
| The next few lines use `pd.concat()` to concatenate the one-hot encoded DataFrames with the original 'df' DataFrame. The original categorical columns are dropped using `df.drop()`, and the resulting DataFrame is assigned back to 'df'. This process is repeated for the 'workclass', 'education', 'marital-status', 'relationship', 'race', and 'native-country' columns. | |
|  | |
| df['gender'] = df['gender'].apply(lambda x : 1 if x =='Male' else 0)`: This line converts the 'gender' column into binary values using a lambda function. Males are encoded as 1, and all other values (presumably females) are encoded as 0.  df['income'] = df['income'].apply(lambda x : 1 if x =='>50K' else 0)`: This line converts the 'income' column into binary values using a lambda function. Income values greater than 50K are encoded as 1, and all other values (income less than or equal to 50K) are encoded as 0. | |
|  | |
| plt.figure(figsize=(18, 12)): This line initializes a new Matplotlib figure with a specified size of 18 inches in width and 12 inches in height. This is done to create a larger plot, making it easier to visualize and analyze the heatmap.  sns.heatmap(df.corr(), annot=False, cmap='coolwarm'): This line creates the heatmap using the Seaborn library. The `df.corr()` function calculates the correlation matrix for the DataFrame 'df', which includes the one-hot encoded features. The `annot` parameter is set to `False`, meaning that the individual correlation values will not be shown on the heatmap. The `cmap` parameter is set to 'coolwarm'. | |

* 1. **Analysis**
     1. **Results of the machine learning model(s)**

|  |
| --- |
| Accuracy Score  The value “0.8639574163169209” means that the optimized Random Forest Classifier model has an accuracy of approximately 86.40% on the test dataset. The high accuracy score suggests that the model able to generalize well and make accurate predictions on the test data.  This score is important because it gives an unbiased estimate of how the model would perform on new, unseen data in a real-world deployment scenario.  Overall, this output indicates that the hyperparameter tuning process was effective, and the resulting Random Forest Classifier model has strong predictive performance on the test set. |

* + 1. **Evaluation metrics used (e.g. accuracy, precision, recall)**

**Correlation**

**Absolute Correlation**

1. Calculate the absolute correlation between each feature and the target "income".

2. Sort the correlation values from low to high.

3. Determine the number of features to drop (80% of total features).

4. Drop the least correlated features from the original dataframe.

**Potentially Relationships**

**•** Spotting any multicollinearity that might be present in the data.

• Guide feature selection and engineering process.

* + 1. **Visualization of results (if applicable)**
  1. **Challenges and Limitations**

Difficulties for beginners: Writing Python can be challenging, especially for beginners. This difficulty arises from the occurrence of errors and the inability to perform calculations correctly. However, the key is to gradually find solutions to these problems once they are identified.

Time-consuming process: Writing Python code often requires investing a significant amount of time. Relying on online resources to find code examples and solutions adds to the time spent on the back-and-forth between writing code and searching for assistance.

Continual updates and testing: During the development process, it is necessary to allocate additional time for testing the program's functionality. Rectifying these problems requires recreating and modifying the code, which prolongs the time.

* + 1. **Discussion of challenges faced during the project**

Lack of expertise: If there is limited understanding of machine learning concepts and techniques, it can be difficult to comprehend and apply appropriate machine learning algorithms and techniques. This can lead to choosing unsuitable models or feature engineering methods, affecting the accuracy of predictions.

Data cleaning and preprocessing: The data may contain missing values, outliers, duplicates, or inconsistent data, which need to be processed and cleaned. For missing values, it is necessary to select appropriate handling methods, such as imputation or removal of samples containing missing values. This process may require some expertise and experience.

Feature selection and construction: Selecting appropriate features is crucial for the performance of the model. Without sufficient domain knowledge or experience in feature engineering, it may be challenging to identify and construct predictive features. This can result in poor model performance or overfitting.

Hyperparameter tuning: Machine learning models often have important hyperparameters that need to be adjusted, such as learning rate, regularization parameters, etc. For beginners, it can be difficult to determine the optimal values for these hyperparameters, requiring multiple experiments and cross-validation to find the best settings.

Model evaluation and interpretation: Understanding evaluation metrics and interpreting the results of the model can be a challenge. For example, understanding the meaning of metrics like accuracy, precision, recall, etc., as well as interpreting the model's predictions and the importance of features.

* 1. **Future Work**
     1. **Description of any additional ideas or approaches that were not implemented**

Explore more advanced techniques and address dataset biases

* + 1. **Reasons for not implementing these ideas (e.g. time constraints, complexity)**

**Potential areas for future improvement**

Practical Use Cases

* 1. **Conclusion**

Key takeaways and insights gained.

**Income Level Prediction**

Government: Understand income distribution, develop targeted policies

Businesses: Profile potential customers, devise effective marketing strategies Individuals: Assess income potential, take appropriate career development measures

**Decision Support**

Government: Adjust education, training, and other policies to improve low-income groups Businesses: Tailor products/services based on model predictions for customer segments Individuals: Use model suggestions to choose more suitable career paths

**Business Insights**

Identify key factors influencing individual income levels.

Understand differences in income-impacting factors across population groups Provide insights for developing more targeted policies or business strategies.

* 1. **Reference**

Kaggle

[www.kaggle.com/datasets/wenruliu/adult-income-dataset/data](http://www.kaggle.com/datasets/wenruliu/adult-income-dataset/data)

iThelp

<https://ithelp.ithome.com.tw/m/articles/10272586>

Tree-Based Model Explained

<https://youtu.be/0VjPG1rmxoc?si=-y6ZUTAUuPI18zvS>

Hyperparameter Tuning and Optimization

<https://medium.com/linux-backend-notes/hyperparameter-tuning-and-optimization-a3368a1370f7>

* 1. **Distribution of Work**

Data research: Dawn,Eric,Jeff

Data cleaning: Dawn,Eric,Jeff

Project report: Dawn,Eric,Jeff