**REPORT**

**Class**: AI17C

**Subject**: DBM302m

**Instructor**: Nguyen Van Vinh – VinhNV27

**Group**: 1

**Members**:

Ha Khai Hoan - QE170157

Dang Phuc Bao Chau - QE170060

Nguyen Van Thu – QE170147

1. **Which problem are you trying to solve why?**



We are interested in understanding which socio-economic factors most influence an individual's income. Specifically, I would like to explore the relationship between factors such as age, education, occupation, and gender in predicting whether an individual will earn more than $50,000 per year. Additionally, I would like to explore whether there are significant income differences between genders and races or orther feature.

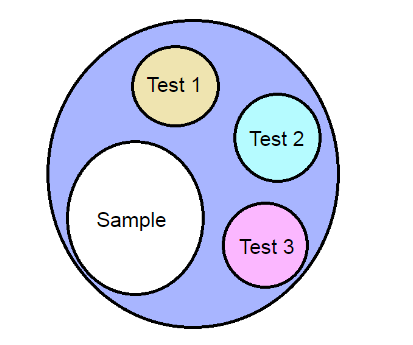
This would be helpful in areas such as:

* **Advertising and marketing**: Companies can target high or low-income groups to offer suitable products or services.
* **Credit analysis**: Financial institutions and banks can use this information to assess repayment ability, plan loans, and set credit limits.
* **Customer segmentation**: Businesses can divide customers by income to develop tailored business strategies, thereby increasing sales efficiency.
* **Public policy**: Government agencies can use this data to shape social policies, such as welfare support for low-income households.
* **Insurance**: Insurance companies can assess risks or design insurance packages tailored to different income groups.
* **Real estate**: Real estate brokers can use this information to predict housing demand among high or low-income earners.
* **Education**: Educational institutions can offer scholarships or support programs based on income levels.

1. **Where and how do you obtain the data? How big is your data?**

We took the Adult Census Income dataset on Kaggle, which is a popular dataset often used to build machine learning models that predict individual income based on demographic factors.

In reality, we do not process and train machine learning models on a huge amount of data. Therefore, we chose the method of selecting samples on a huge population, then conducting testing on the selected sample, to determine the right path when working on that huge amount of data.



*The selected data will be a sample, it represents a huge data set, to conduct research to find hidden characteristics and values, instead of using all the characteristics, we will use the important characteristics, thereby reducing resource costs, improving speed when deploying on large data sets.*

+ Number of rows of data: 32561

+ Number of columns of data: 15

Note:

|  |  |  |
| --- | --- | --- |
|  | **Feature** | **Description** |
| 1 | Age | Describes the age of individuals. Continuous. |
| 2 | Workclass | Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked. |
| 3 | fnlwgt | Continuous. A weighting factor created by the US Census Bureau indicating the number of people represented by each data entry. |
| 4 | Education | Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, Preschool. |
| 5 | Education-num | Number of years spent in education. Continuous. |
| 6 | Marital-status | Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse. |
| 7 | Occupation | Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, etc. |
| 8 | Relationship | Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried. |
| 9 | Race | White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black. |
| 10 | Sex | Female, Male. |
| 11 | Capital-gain | Represents the profit from the sale of assets (e.g., stocks or real estate). Continuous. |
| 12 | Capital-loss | Represents the loss from the sale of assets (e.g., stocks or real estate). Continuous. |
| 13 | Hours-per-week | Continuous. |
| 14 | Native-country | List of countries including United-States, Cambodia, England, Puerto-Rico, Canada, Germany, etc. |
| 15 | Salary | >50K, <=50K. |

1. **What are your ideas to solve the problem?**

My approach is to apply various machine learning classification algorithms such as:

+ Logistic Regression for its simplicity and interpretability.

+ Random Forest for handling non-linear relationships and importance weighting of features.

+ KNN (K-Nearest Neighbors) is a supervised learning algorithm.

The pipeline will include:

+ Data preprocessing:

. Missing handle

. Duplicate handle

. Outlier handle

+ Feature engineering:

Separate categorical and numerical features for easy management.

* Categorical features
* Numerical features

+ Build model

+ Model tuning to optimize performance.

In addition, I also visualized the data to better understand the interactions between features, to identify which groups of factors are important in predicting whether a person is truly high-income or not.

1. **What is your hypothesis for the ideas to work? A more interesting question is how do you verify your hypothesis?**

Hypothesis: Certain features such as **education, age, and occupation** will have the strongest predictive power for determining income. I hypothesize that more educated individuals or those in higher-tier occupations are likely to earn more than 50K USD.

To verify this, I will:

+ Conduct exploratory data analysis (EDA) to check feature distributions.

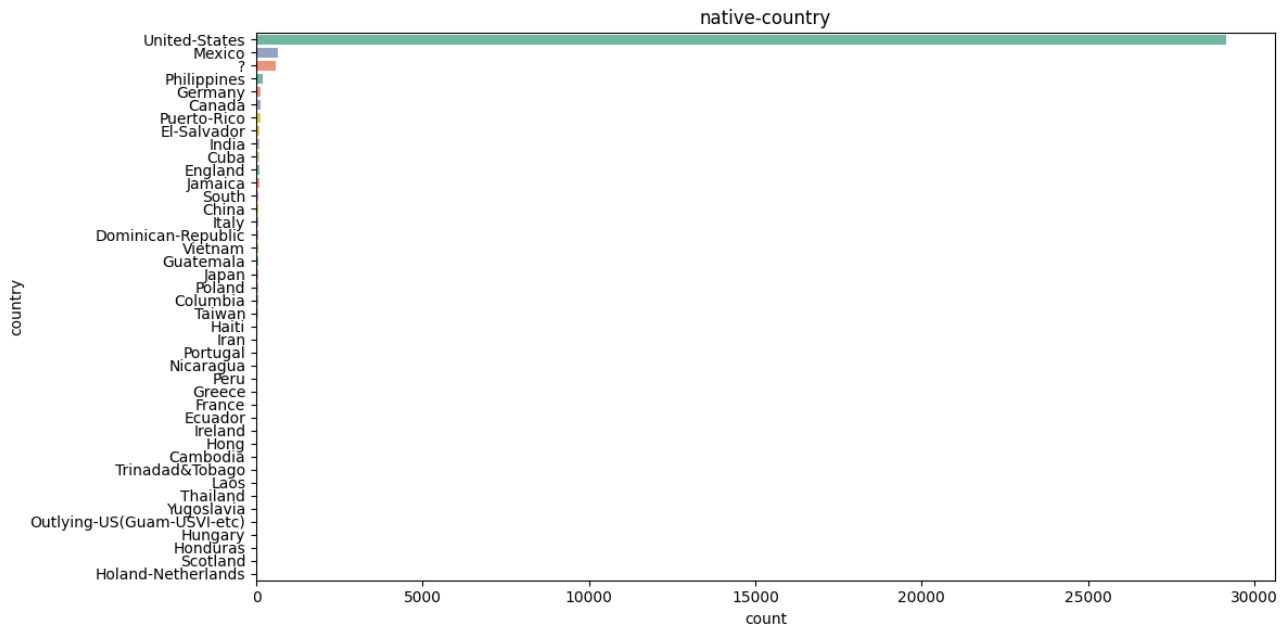
+ Use feature importance analysis from Random Forest,….

+ Compare model performance through accuracy, **precision**, recall, and F1-score on a test dataset.

+ Validate the models with cross-validation to ensure generalizability.

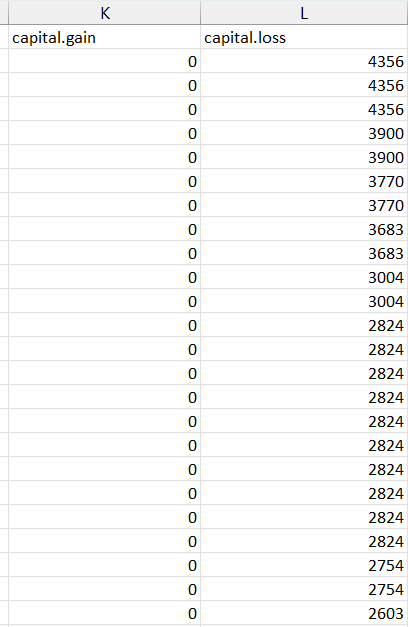
1. **How does the result look like? Does it confirm your hypothesis?**

**Data processing**

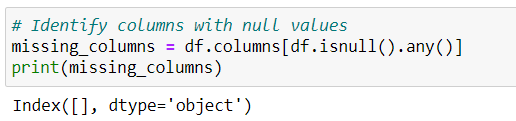
****

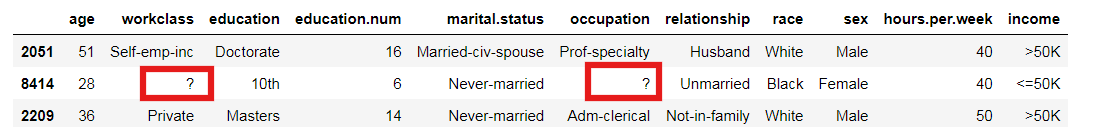
+ We removed columns that did not have values ​​or caused noise in the prediction results.

* `fnlwgt`: This is a weight created by the US Census Bureau to represent the number of people each sample represents. While it may be statistically significant in census research, it generally has little impact in predicting income.
* `capital-gain`, `capital-loss`: Contains too many zeros, the analysis will not make sense.

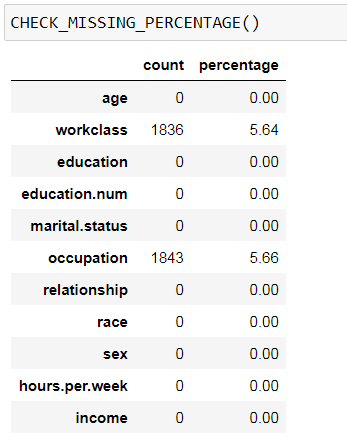


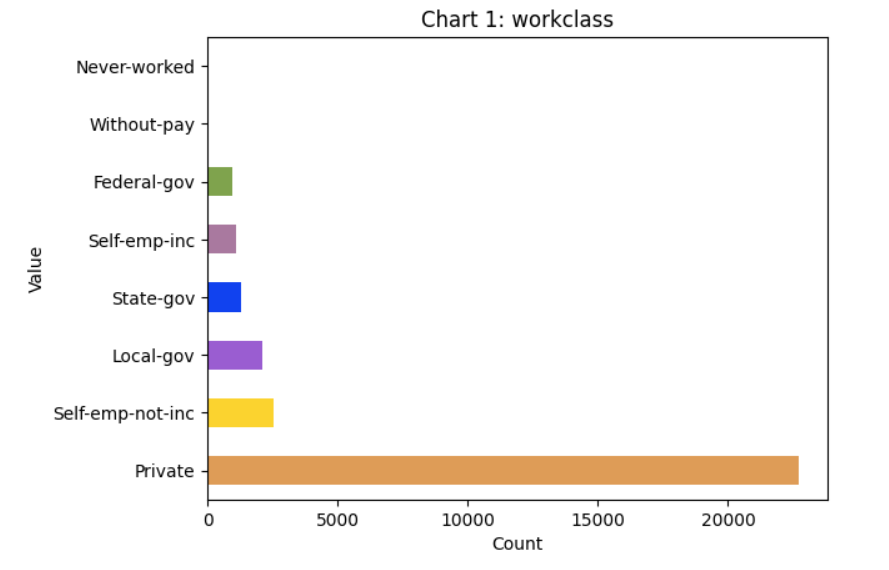
* `native-country`: This is a value column that shows a person's original nationality, but they are living in the same country as the United States, all income is calculated in US currency and assets, and the majority of people with US origin nationality are, so analyzing this column is also not valuable.

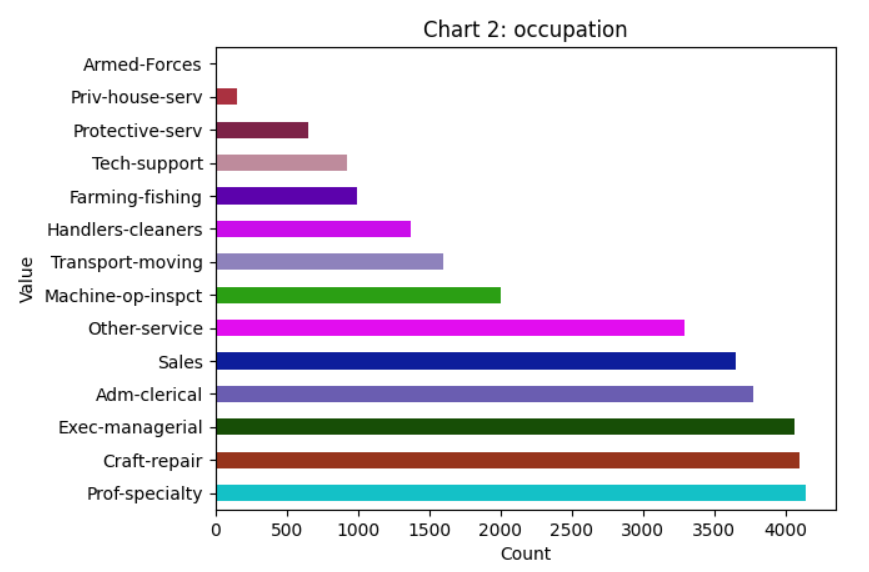




+ We have checked the dataset, we found, there is no value representing "null", but in the table there are a lot of "?", so this is a `null` value, filled in as a character. We have worked hard to solve them.

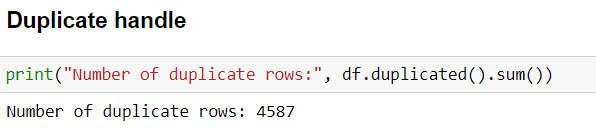




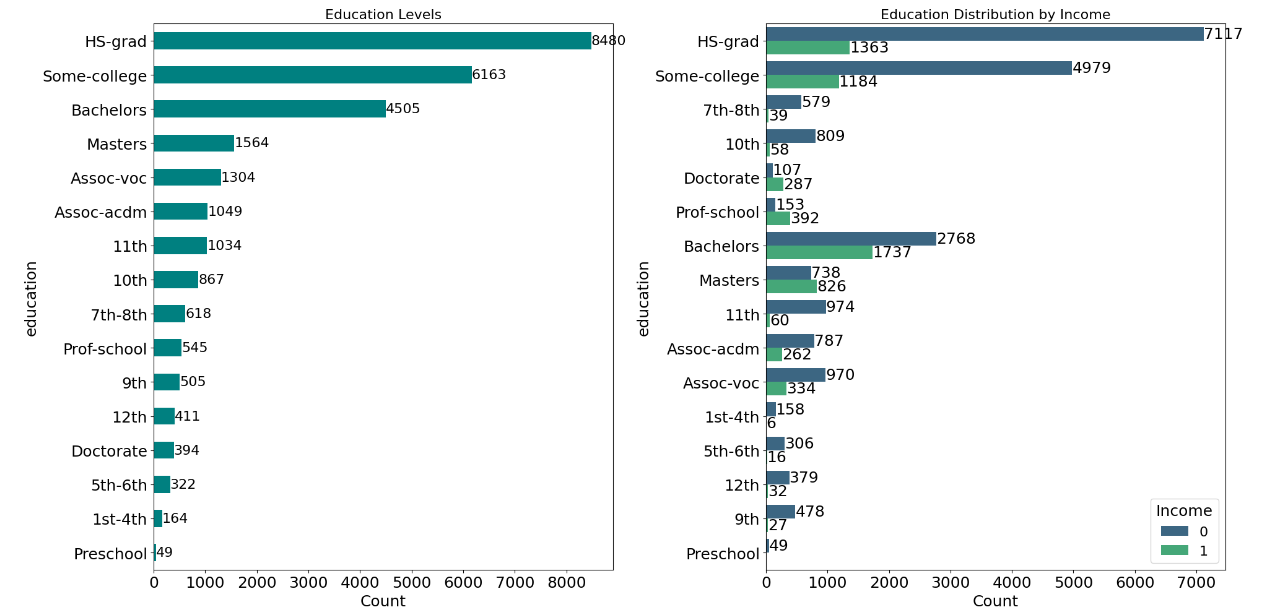


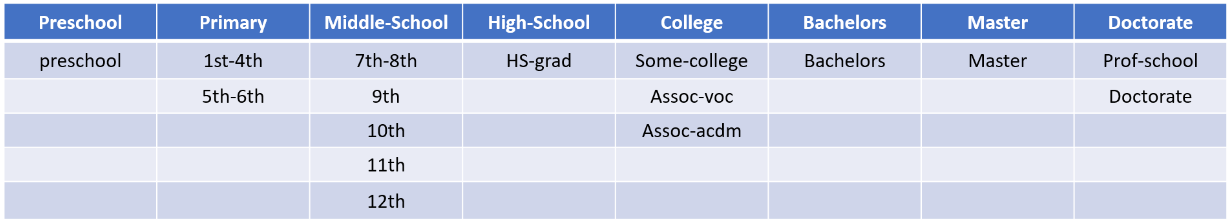
* So with charts 1, we can see that the percentage of numbers skewed to one column is very high, so with these two tables, we will fill in the most common values ​​in the cells.
* With chart 2, we will spread the values ​​evenly into the cells.

+ We remove duplicate rows.

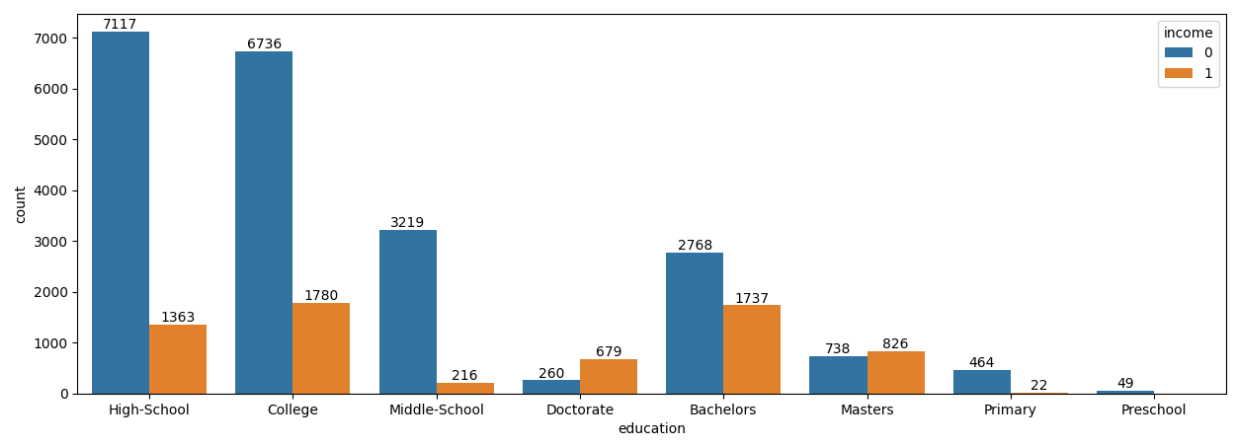


+ Put the discrete values ​​in the same range so as not to disturb the model.





Result:



**Conduct hypothesis testing.**

***We have a data set, so we will process the data, then divide our data into 4 parts, with 1 part for training, the remaining 3 parts for testing, to observe the improvement of prediction results through each step.***

* Train set: 21000 rows
* Test set 1: 2000 rows
* Test set 2: 2000 rows
* Test set 3: 2974 rows

I used 3 features `education`, `age`, and `occupation` as input for the models, with 'income' as the output of the model.

We have chosen the Precision measure of income > 50k USD (here we set it to 1) to evaluate the quality of the hypothesis and the actual needs. We are interested in the number of correct predictions of 1 compared to all predictions that the model gives is 1, this has great significance for accurately classifying subjects with high income, serving the needs of borrowing money at the bank, financial inventory,... in the most accurate way.

The results after training the model and testing were not really promising, it showed that the hypothesis I initially put forth was not true to reality.

+ On Test set 1:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Precision (0)** | **Precision (1)** | **Recall (0)** | Recall (1) | **F1-Score (0)** | **F1-Score (1)** | **Accuracy** | **Macro Avg F1** | **Weighted Avg F1** | **Train Time (s)** | **Infer Time (s)** |
| **Logistic Regression** | 0.76 | **0.36** | 0.97 | 0.05 | 0.85 | 0.09 | 0.75 | 0.56 | 0.67 | 0.0225 | 0.001 |
| **Random Forest** | 0.81 | **0.51** | 0.9 | 0.32 | 0.85 | 0.4 | 0.76 | 0.66 | 0.74 | 1.3788 | 0.0524 |
| **KNN** | 0.81 | **0.50** | 0.89 | 0.35 | 0.85 | 0.41 | 0.76 | 0.66 | 0.74 | 0.0141 | 0.0728 |

+ On Test set 2:

|  |  |  |
| --- | --- | --- |
| **Model** | **Precision (1)** | **Infer Time (s)** |
| **Logistic Regression** | **0.43** | 0 |
| **Random Forest** | **0.51** | 0.0457 |
| **KNN** | **0.47** | 0.0563 |

+ On Test set 3:

|  |  |  |
| --- | --- | --- |
| **Model** | **Precision (1)** | **Infer Time (s)** |
| **Logistic Regression** | **0.39** | 0 |
| **Random Forest** | **0.51** | 0.0548 |
| **KNN** | **0.50** | 0.0692 |

1. **What have you done to make your original ideas better?**

In order to check the quality of the models we selected, we took all the features present in the processed data and put them into the model.

'age', 'workclass', 'education', 'education.num', 'marital.status',  'occupation', 'relationship', 'race', 'sex', 'hours.per.week','income'.

Still keep the same 3 old models.

The results are quite promising.

+ On Test set 1:

|  |  |  |
| --- | --- | --- |
| **Model** | **Precision (1)** | **Infer Time (s)** |
| **Logistic Regression** | **0.63** | 0 |
| **Random Forest** | **0.58** | 0.0547 |
| **KNN** | **0.60** | 0.2285 |

+ On Test set 2:

|  |  |  |
| --- | --- | --- |
| **Model** | **Precision (1)** | **Infer Time (s)** |
| **Logistic Regression** | **0.60** | 0 |
| **Random Forest** | **0.53** | 0.0583 |
| **KNN** | **0.57** | 0.22 |

+ On Test set 3:

|  |  |  |
| --- | --- | --- |
| **Model** | **Precision (1)** | **Infer Time (s)** |
| **Logistic Regression** | **0.62** | 0.0012 |
| **Random Forest** | **0.60** | 0.0795 |
| **KNN** | **0.59** | 0.4569 |

Moving on, we applied two more complex algorithms, XGBoost and SVM (Support Vecto Machine).

The results improved quite a bit.

+ On Test set 1:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Precision (0)** | **Precision (1)** | **Recall (0)** | **Recall (1)** | **F1-Score (0)** | **F1-Score (1)** | **Accuracy** | **Macro Avg F1** | **Weighted Avg F1** | **Train Time (s)** | **Infer Time (s)** |
| **XGBoost** | 0.86 | **0.65** | 0.91 | 0.53 | 0.88 | 0.58 | 0.82 | 0.75 | 0.81 | 0.18 | 0.01 |
| **Support Vector Machine** | 0.85 | **0.70** | 0.93 | 0.48 | 0.89 | 0.57 | 0.82 | 0.77 | 0.81 | 10.2629 | 1.4866 |

+ On Test set 2:

|  |  |  |
| --- | --- | --- |
| **Model** | **Precision (1)** | **Infer Time (s)** |
| **XGBoost** | **0.64** | 0.01 |
| **Support Vector Machine** | **0.65** | 1.7867 |

+ On Test set 3:

|  |  |  |
| --- | --- | --- |
| **Model** | **Precision (1)** | **Infer Time (s)** |
| **XGBoost** | **0.67** | 0.018 |
| **Support Vector Machine** | **0.68** | 2.1215 |

1. **What is the running time of your algorithm? Is your algorithm scalable?**

The algorithms we use are machine learning algorithms, from simple to complex, with simple algorithms having very fast training and deployment time (Logistic Regression, Random Forest, KNN), giving relatively poor results, with complex algorithms (XGBoost, SVM) having very slow training and deployment time, but in return, the results are better, especially with SVM.

We have used PCA, applying different data normalization methods, but the results obtained are that the metrics are simplified, the deployment time is longer.

*from sklearn.preprocessing import StandardScaler, LabelEncoder*

*label\_encoders = {}*

*for column in ['workclass', 'education', 'marital.status', 'occupation', 'relationship', 'race', 'sex']:*

*le = LabelEncoder()*

*hypothesize\_df[column] = le.fit\_transform(hypothesize\_df[column])*

*label\_encoders[column] = le*

*scaler = StandardScaler()*

*------------------------------------------------------*

*from sklearn.preprocessing import OneHotEncoder, StandardScaler*

*from sklearn.decomposition import PCA*

*scaler = StandardScaler()*

*preprocessor = ColumnTransformer(transformers=[*

*('num', scaler, numerical\_columns),*

*('cat', onehot\_encoder, categorical\_columns)*

*])*

*X\_train\_encoded = preprocessor.fit\_transform(X\_train)*

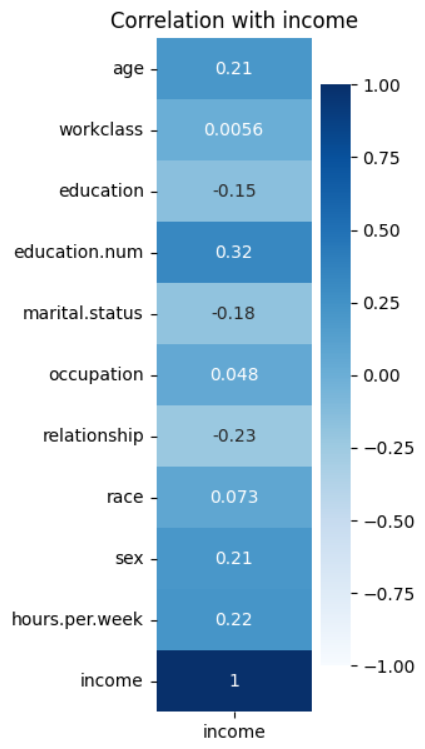
*X\_test\_encoded = preprocessor.transform(X\_test)*

PCA may only be suitable for data with thousands to millions of features to see the effectiveness clearly, because this algorithm has to go through complex steps of calculating the covariance matrix.

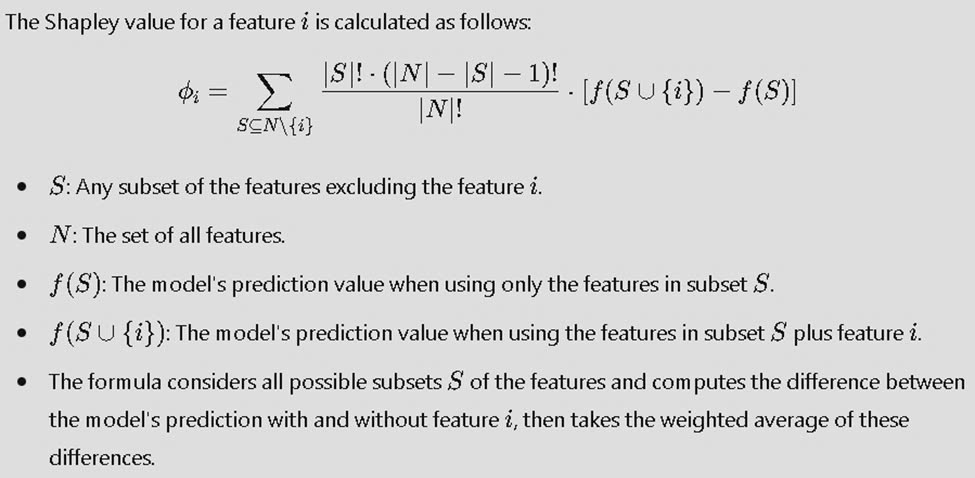
We use a test set, that is a combine set [[Test set 1] + [Test set 2] + [Test set 3]]

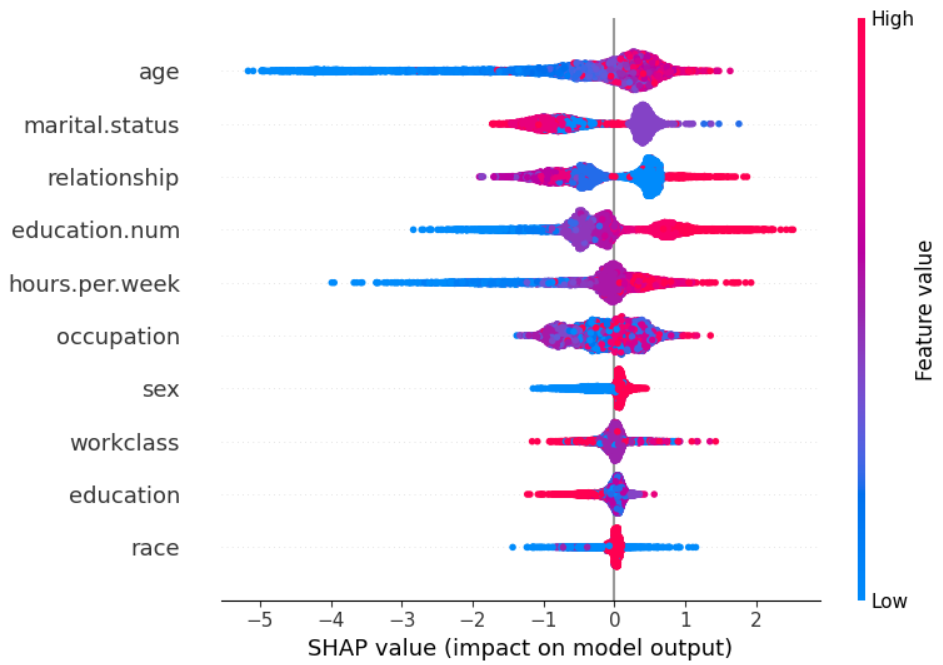
|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Precision (0)** | **Precision (1)** | **Recall (0)** | **Recall (1)** | **F1-Score (0)** | **F1-Score (1)** | **Accuracy** | **Macro Avg F1** | **Weighted Avg F1** | **Train Time (s)** | **Infer Time (s)** |
| **XGBoost (PCA)** | 0.86 | **0.61** | 0.9 | 0.51 | 0.88 | 0.56 | 0.81 | 0.72 | 0.8 | **0.1618** | **0.0071** |
| **SVM (PCA)** | 0.85 | **0.65** | 0.92 | 0.5 | 0.88 | 0.56 | 0.82 | 0.72 | 0.81 | **6.636** | **3.6472** |

We used another approach to improve speed while reducing the dimensionality of the data.

We calculated correlations for the features in the data.

And calculate important features based on Shap (SHapley Additive exPlanations) algorithm, implemented on XGBoost. (**Machine Learning Interpretability**)





Based on the results of the correlation plot and the shap plot on the Xgboost results, I found the features that have strong correlation with "income", they are 'age', 'relationship', 'education.num', 'hours.per.week', 'sex'.

* `+1`: Perfect positive linear relationship.
* `-1`: Perfect negative linear relationship.

+ On Test set 1:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Precision (0)** | **Precision (1)** | **Recall (0)** | **Recall (1)** | **F1-Score (0)** | **F1-Score (1)** | **Accuracy** | **Macro Avg F1** | **Weighted Avg F1** | **Train Time (s)** | **Infer Time (s)** |
| **XGBoost** | 0.86 | **0.65** | 0.91 | 0.53 | 0.88 | 0.59 | 0.82 | 0.75 | 0.81 | **0.1272** | **0.005** |
| **Support Vector Machine** | 0.84 | **0.72** | 0.95 | 0.43 | 0.89 | 0.54 | 0.82 | 0.78 | 0.81 | **8.3188** | **1.5922** |

+ On Test set 2:

|  |  |  |
| --- | --- | --- |
| **Model** | **Precision (1)** | **Infer Time (s)** |
| **XGBoost** | **0.63** | **0.005** |
| **Support Vector Machine** | **0.68** | **1.4351** |

+ On Test set 3:

|  |  |  |
| --- | --- | --- |
| **Model** | **Precision (1)** | **Infer Time (s)** |
| **XGBoost** | **0.64** | **0.007** |
| **Support Vector Machine** | **0.7** | **1.9893** |

The results show that, although I have reduced the number of features significantly, the results are improved due to the removal of noisy features, while accurately identifying important features needed for "income".

    + Improve the speed of training, and testing.

    + Improve the accuracy of target metrics.

Scalability: While Logistic Regression and Random Forest are scalable, XGBoost is designed to handle large datasets efficiently due to its tree-boosting algorithm, making it the most scalable for big data problems. SVM, though powerful for smaller datasets, may struggle with scalability on larger datasets due to its computational complexity, unless optimizations like linear SVM or kernel approximations are used.

1. **If you are given more time, what can be done to even improve it further?**

If given more time, I would:

+ Experiment with deep learning models like neural networks to see if they outperform traditional machine learning methods.

+ Perform additional feature engineering, especially for categorical variables.

+ Explore ensemble methods by combining multiple models for better accuracy.

+ Perform more advanced hyperparameter tuning using Bayesian optimization.

1. **What have you learned from the project?**

Data Preprocessing: I learned the importance of handling categorical and numerical data differently. Encoding categorical variables and scaling numerical features were crucial steps in preparing the data for algorithms like SVM, Logistic Regression, and Random Forest.

Model Selection: I explored various machine learning models, each with its strengths and limitations. For example, Logistic Regression worked well for binary classification but struggled with non-linear relationships. Random Forest showed better performance for imbalanced datasets but required more computational time. SVM provided high accuracy but was slower on larger datasets due to computational complexity.

Scalability Concerns: Models like XGBoost and Random Forest demonstrated their efficiency with larger datasets, while SVM had limitations in terms of scalability, which highlighted the need for choosing algorithms based on dataset size and computational resources.

Select important features, remove unimportant features in the data so that the model gives the best prediction results.

Trade-off between time and accuracy, with simple algorithms, we have short execution time, but low accuracy, and with complex algorithms, we have long execution time, but in return we get better quality results.

---The End---