Computer Science Department

Final Project: Income Classification Problem

Course: Introduction to AI (CS156)

Course Instructor: Yulia Newton, Ph.D.

Ricky Kuang

Introduction/Background

The problem that I will be attempting to solve is to see what factors potentially affect people's annual incomes. There have been many debates over income inequality throughout the entire world, so I thought it would be interesting to analyze what factors potentially play a factor in someone's income. The prediction problem is to predict whether a person's annual income is $50,000 or less, or if they make over $50,000 depending on factors. Therefore, this is a binary classification problem that I am attempting to solve; one class being annual income less than or equal to $50K and the other being annual income is greater than $50K.

It's important to solve this problem because it will help explain what affects the income of an individual, which will help point us to potential solutions to increase overall income or putting more focus on areas like education if that is an influential factor on annual income. There have been previous attempts to solve this problem, as I've noticed others using methods like KNN or decision trees to identify what factors may affect annual income. I plan to take a similar approach for this term project by taking factors such as education and occupation to try and arrive at a solution to the problem.

Dataset description

The dataset that I am going to use to solve the described problem is the Income Classification dataset that I found on Kaggle: https://www.kaggle.com/datasets/lodetomasi1995/income-classification

The independent variables that I plan to use to train my model are listed in the following table:

Feature	Description
age	The age of a person
workclass	The working class that one belongs to
educational_num	Highest education that a person achieved
marital_status	Whether someone is married or not
occupation	What one does for a living
relationship	A person's familial relationship
race	One's race
gender	Male or female depending on what one identifies as
capital_gain	The amount of capital gain an individual makes.
capital_loss	The amount of capital loss an individual suffers.
hours_per_week	How much one works in a week

The training data is supposed to represent the different factors that affect the level of income of an individual. The columns are a mix of categorical and numerical columns.

The response variable if a categorical variable income_>50K, which indicates whether an individual makes over $50,000 or not. Individuals below $50,000 are represented by 0 and those above are represented by 1. The model that I create will be aiming to predict whether an individual is either 0 or 1 based on the independent variables.

The idea for the data split is to split it 80/20 between training and test data. There will be a total of about 35,165 training observations. The problem is a classification problem, so in the observation training data, there are 26,764 observations of class label 0 ($50,000 or less income) and 8,401 observations of 1 (over $50,000 income).

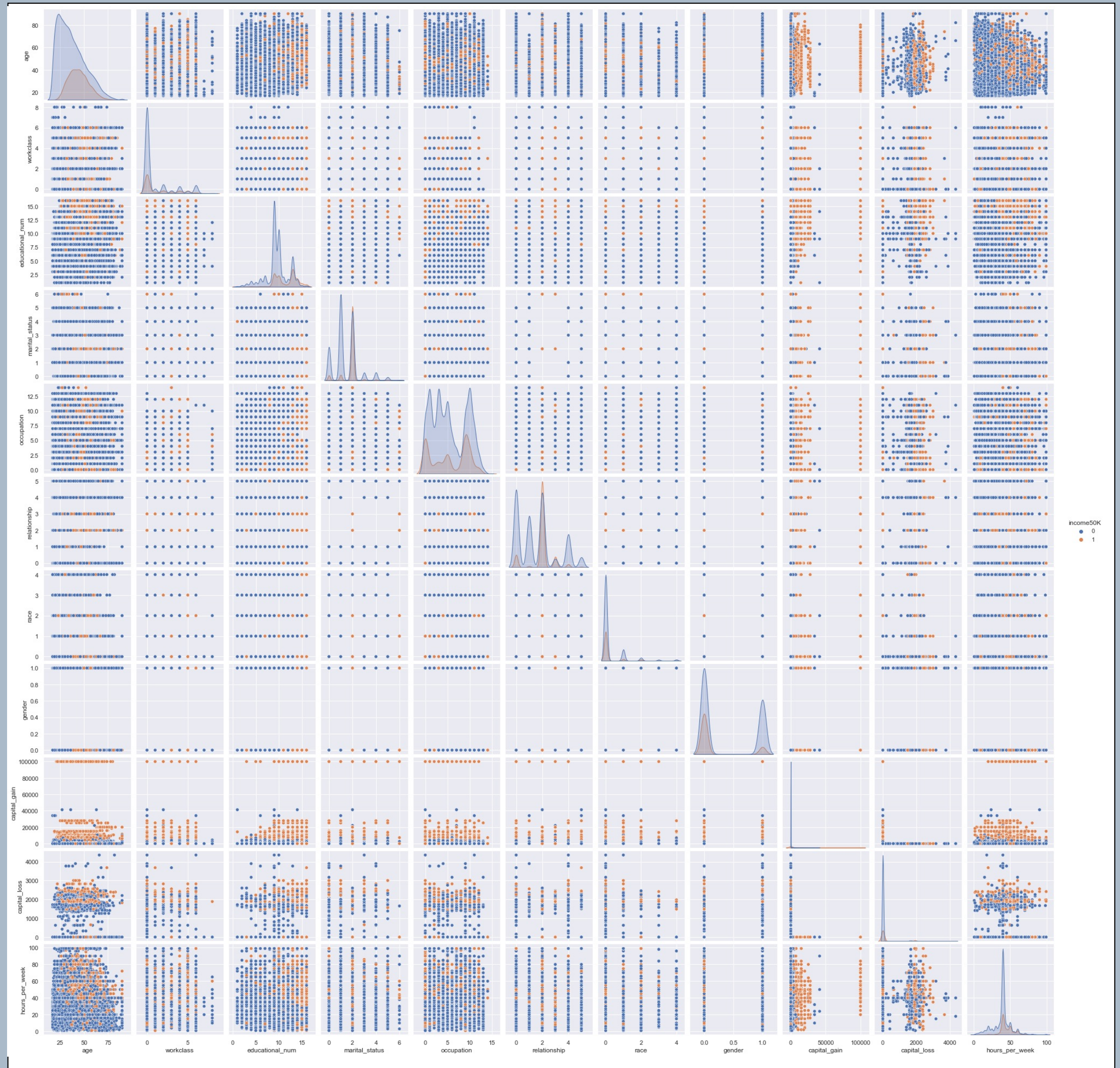


Figure 1: pairplot of all the features of the income classification problem.

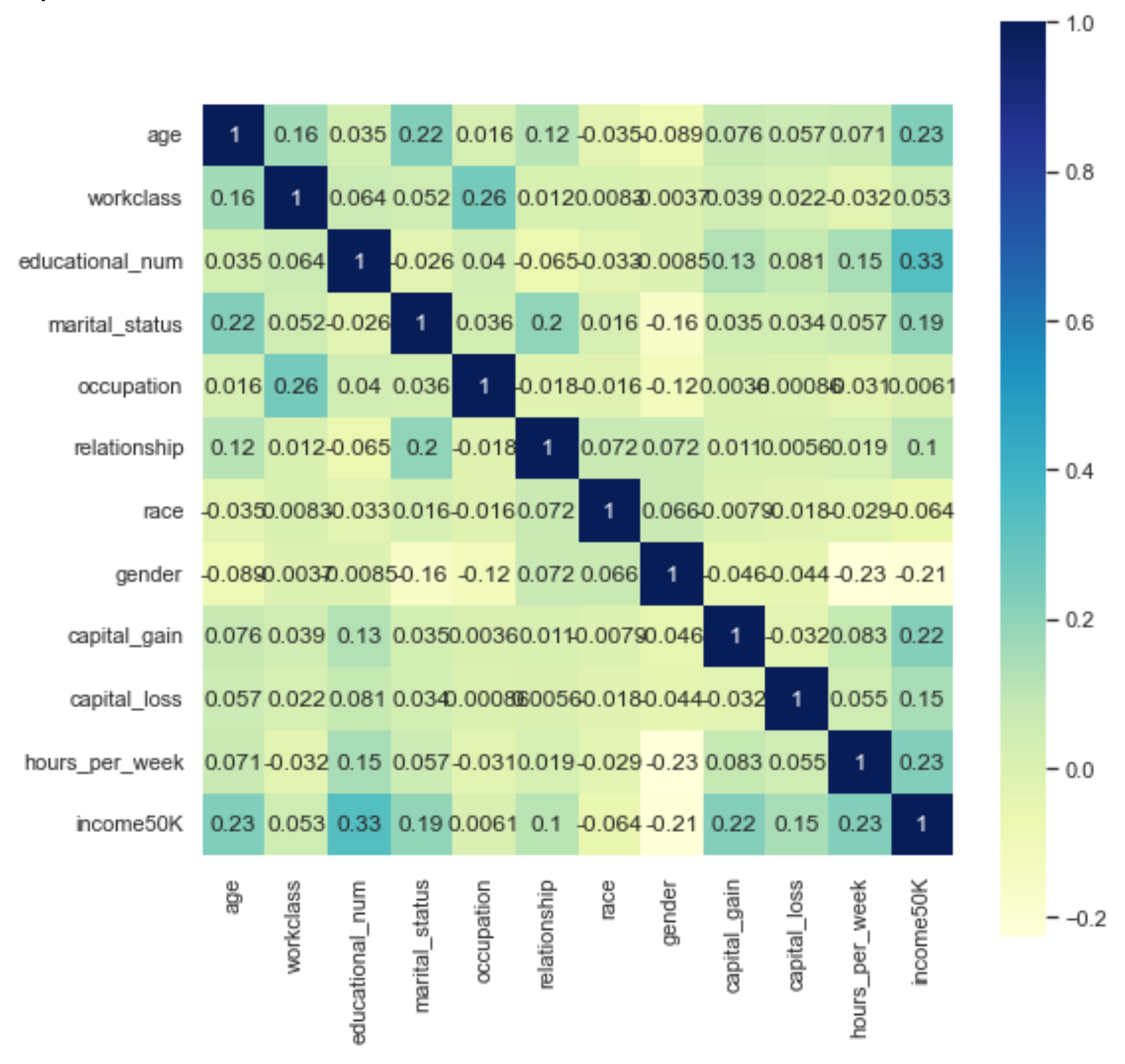


Figure 2: correlation plot showing the relationship between the features.

Methodology

Algorithm

The algorithm that I used for this project is the K Nearest Neighbors algorithm. Since this is a classification problem that aims to see which factors contribute to one's annual income, I felt that running the KNN algorithm on this dataset would help pinpoint whether these factors do in fact play a role in annual income being below or above $50K.

Application to the project

For this project, I dropped some of features as I felt that some were either redundant or didn't contribute much. The following features were dropped in this classification problem:

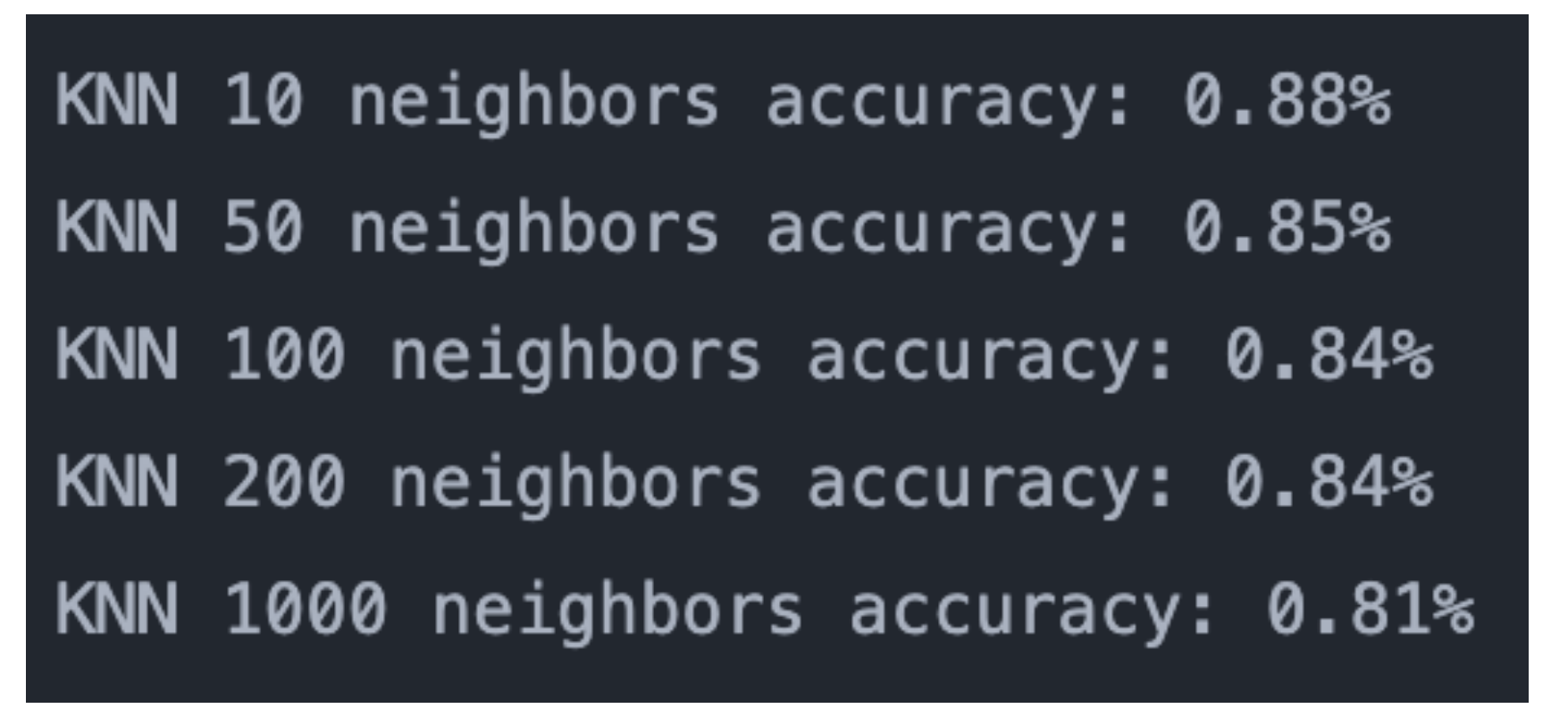
Feature	Reason for dropping
fnlwgt	How many people an entry represents. I didn't think it was necessary for this classification problem.
education	Same info as 'educational_num'
native_country	Other features don't account for countries' status (like what is the economic situation like), so it felt unfair to factor in native country.

For the features that were included (which are listed in the "Dataset description" section), I converted the categorical columns into numeric columns by mapping each value to an integer. I was originally going to use one-hot encoding, but I felt that it added too many columns and it made it harder to visualize data distribution when there were too many features. By mapping categorical values to integers, the size and shape of the data is maintained.

Analysis and Results

Training Results

I trained multiple KNN models with varying neighbors parameters to see whether more or less neighbors would affect the results dramatically. To measure the performance of the KNN models, I used the accuracy score metric from the scikit learn metrics library. After running predictions on the training data through the KNN models, I compared the predicted results to the original training data. The average accuracy for 5 different models is 84.4%. I noticed that the models with a lower amount of neighbors parameter performed better in terms of accuracy.

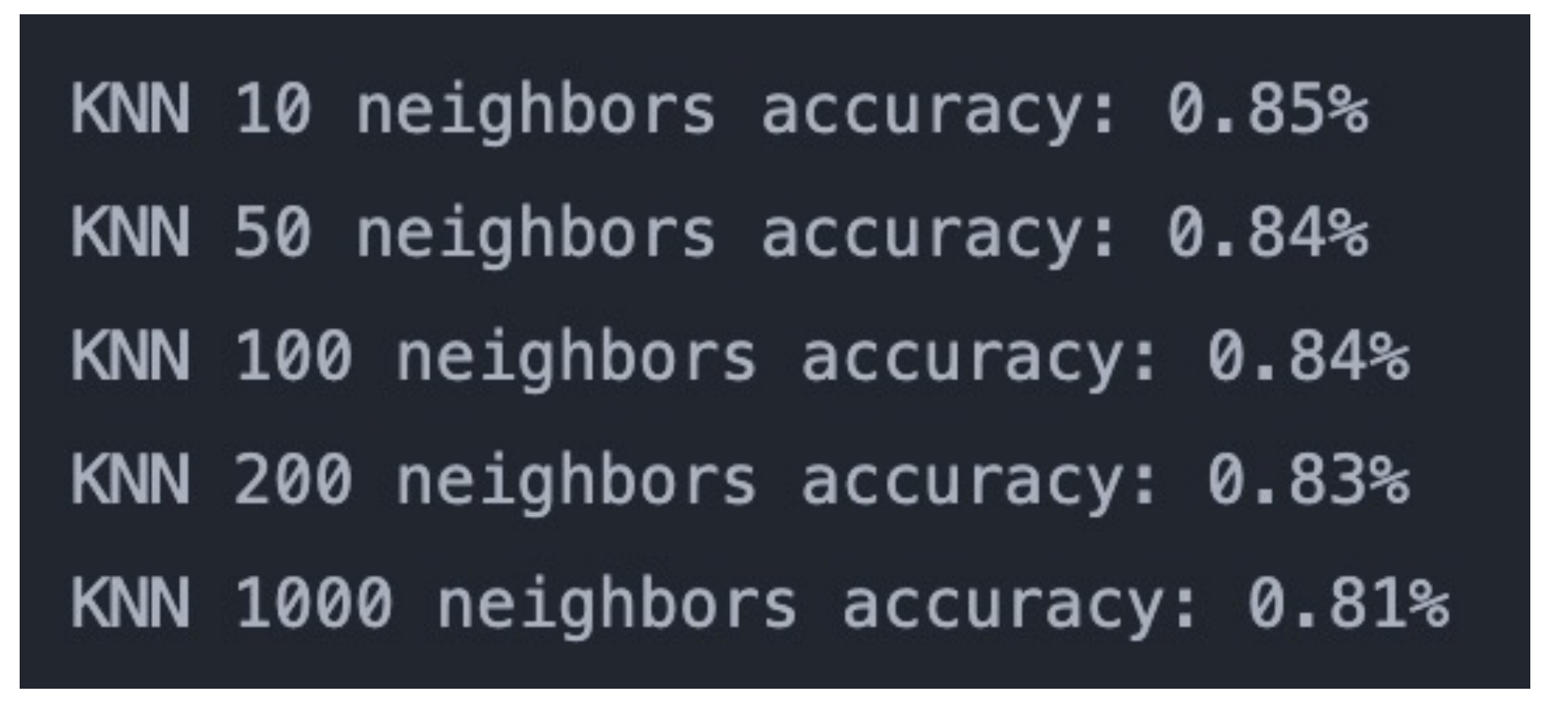


```
KNN 10 neighbors accuracy: 0.88%
KNN 50 neighbors accuracy: 0.85%
KNN 100 neighbors accuracy: 0.84%
KNN 200 neighbors accuracy: 0.84%
KNN 1000 neighbors accuracy: 0.81%
```

Figure 3: accuracy scores for the KNN models when predicting training data

Test results

Using the same KNN models that I trained before, I ran predictions on the set-aside test data. To measure the performances of the KNN models, I used the confusion matrix and accuracy score metrics from the scikit learn metrics library. The average accuracy for 5 different models is 83.4%. Similarly to the results of the KNN models' predictions on the training data, the models with lower amount of neighbors performed better.



```
KNN 10 neighbors accuracy: 0.85%
KNN 50 neighbors accuracy: 0.84%
KNN 100 neighbors accuracy: 0.84%
KNN 200 neighbors accuracy: 0.83%
KNN 1000 neighbors accuracy: 0.81%
```

Figure 4: accuracy scores for each of the KNN models predicting the test data

For the confusion matrices of the KNN models, I noticed that the true positive cells performed better as the number of neighbors increased while the true negative cells performed worse. The true positive cells performed well on all matrices regardless, while the true negative cells performed poorly for each of the matrices. This leads me to believe that the features used may not have been realistic factors for individuals that make above $50K annually.

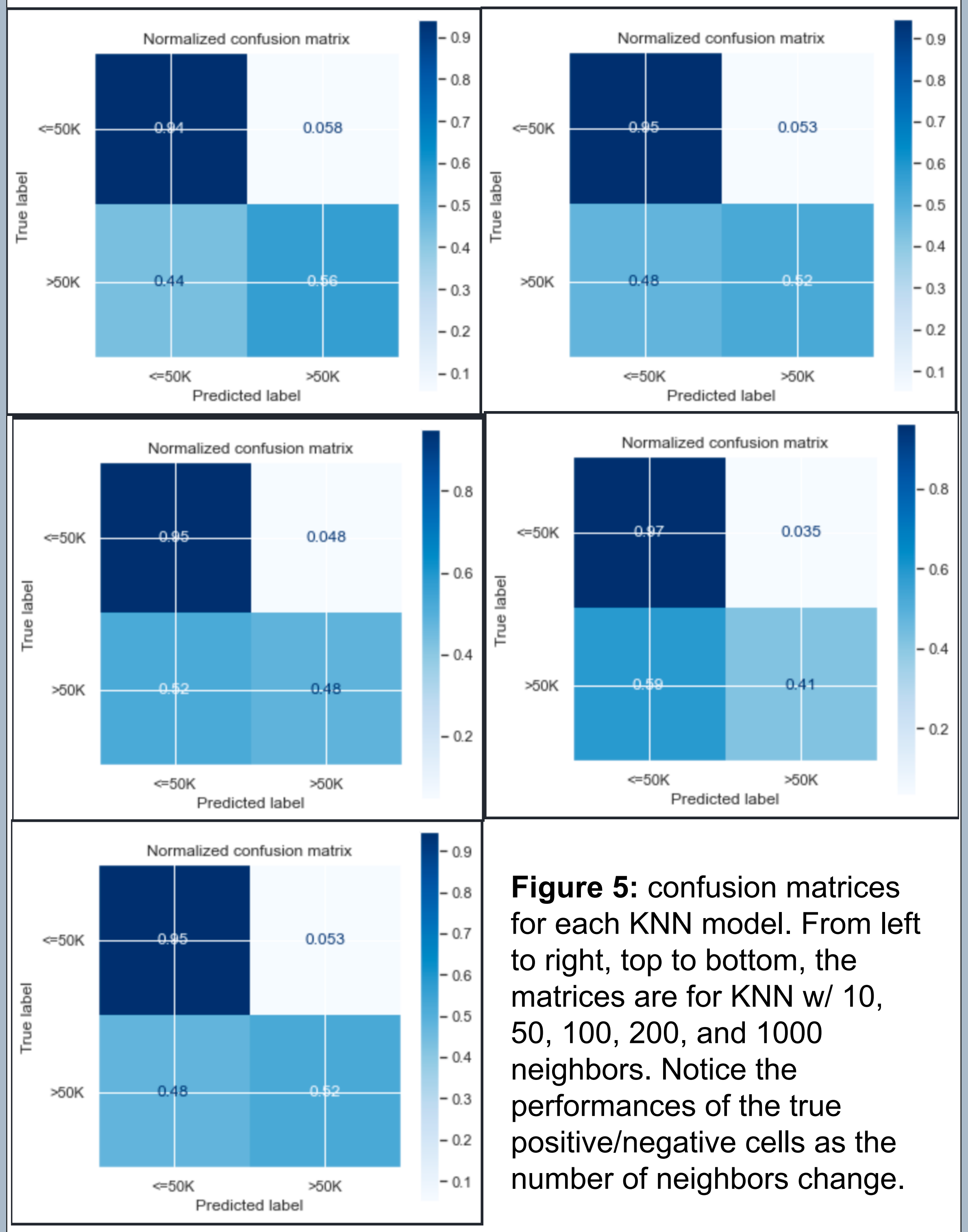


Figure 5: confusion matrices for each KNN model. From left to right, top to bottom, the matrices are for KNN w/ 10, 50, 100, 200, and 1000 neighbors. Notice the performances of the true positive/negative cells as the number of neighbors change.

Summary/Conclusions

On paper, the results of the KNN models performed sufficiently, averaging around 84% accuracy when run on both the training and test data. However, the fact that only the incomes less than or equal to $50K performed well while the incomes greater than $50K performed poorly leads me to believe that the features of the dataset don't accurately represent the latter class. To improve this project, I would include other more relevant features since there are probably some factors that affect income that aren't accounted for in the dataset. I'd also experiment with dropping some of the features of the current dataset. I'm not sure if the results would be better given a different algorithm, but it would be interesting to research the results of other algorithms.

Key References

[1] de Tomasi, Lorenzo, "Income classification" Kaggle Dataset, https://www.kaggle.com/datasets/lodetomasi1995/income-classification?datasetId=149550&searchQuery=scatter.
[2] Newton, Yulia, Jupyter Notebook Examples.
[3] Kumar, Bijay, "Matplotlib increase plot size", PythonGuides, https://pythonguides.com/matplotlib-increase-plot-size/.
[4] "Python | Pandas dataframe.select_dtypes()", GeeksforGeeks, https://www.geeksforgeeks.org/python-pandas-dataframe-select_dtypes/

www.PosterPresentations.com