Machine Learning Adult Wage Prediction

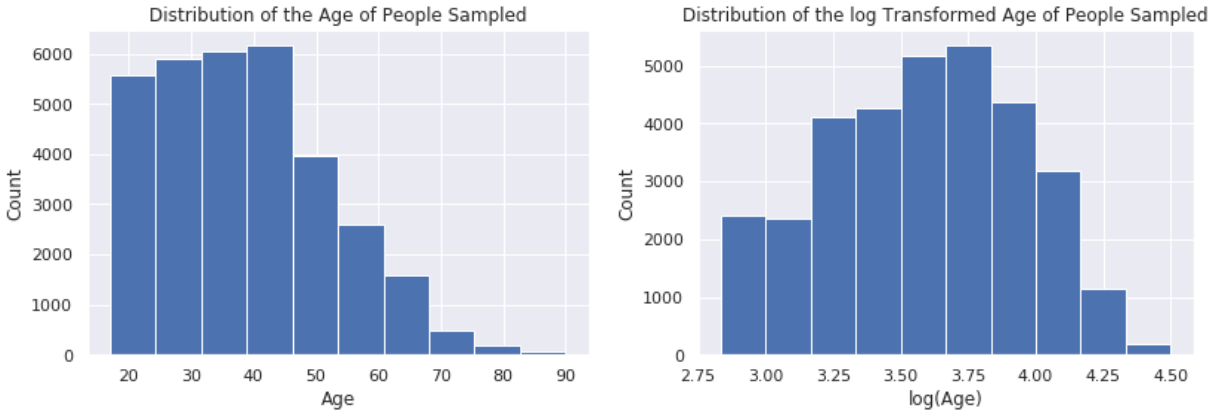
Objective:

We worked with a popular machine learning (ML) data set called the adult wage data set (https://archive.ics.uci.edu/ml/data sets/Adult). The data set consists of the following features for each person: age, work class, weight of sample, education, number of years in education, marital status, occupation, relationship, race, sex, capital gain, capital loss, work hours, and native country. Based on these features, the challenge is to predict whether a person has a salary over US$ 50k. Our objective was to build and compare different models to determine the model with the highest validation score possible (without overfitting), and to explore different techniques which can improve the accuracy of these models.

Data Preprocessing:

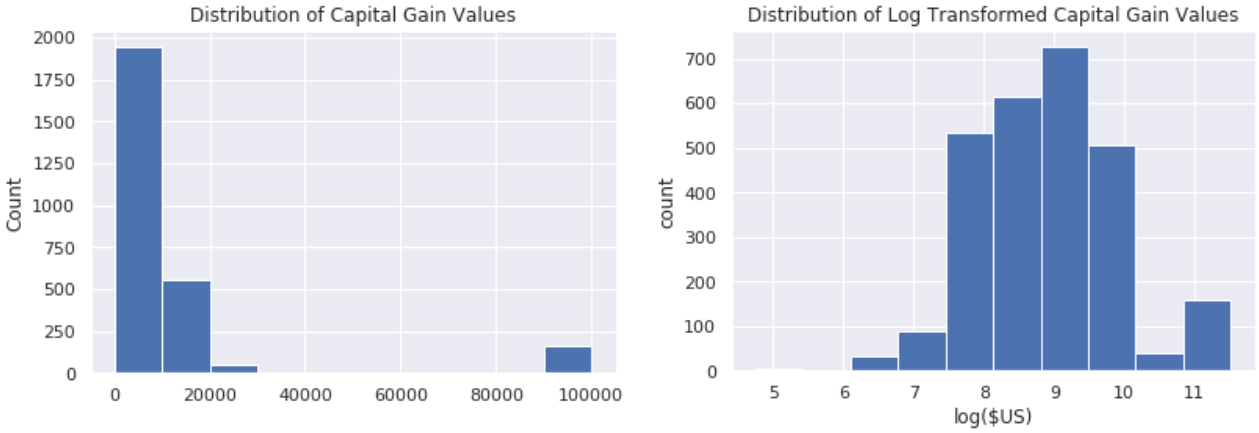
1. **Removing unused features:**
    a. ‘fnlwgt’: sampling weight. We did not know the proper techniques to handle this feature.
    b. ‘education’: categorical data for education level. The information contained in this feature seemed to overlap with the ‘education-num’ feature (the number of years of school attended by a person).
    c. ‘native-country’: this categorical data was too unbalanced (the majority had the value “United-States”), and the information overlapped moderately with the ‘race’ feature.

2. **Scaling numerical features:**
    a. ‘age’ was distributed with a bias to the left. Log transformation was applied to normalize the distribution. This would help some of the ML models, since the transformation evens out the Euclidean distances between data points within this feature, which makes it easier to find boundaries between the categories in Euclidean space. The accuracy of ML models improved slightly as a result.



b. “education-num” values increment linearly. We thought that the number of years in education after graduating highschool should have more weight than the number of years prior, since a highschool diploma is usually needed to get a high-paying job. Hence, we multiplied by a factor of three all the “education-num” values greater than eight (which is the number of years it takes a person to graduate highschool in this data set). This transformation moderately improved our score.

c. ‘capital-gain’ feature was distributed unevenly, largely due to most of the values being zero. Log transformation was applied to even out the distances between data points, similar to the transformation on ‘age’ feature. Note that values of zero were excluded in the histogram figures below to illustrate the point that log transformation more or less produced a normal distribution.



d. ‘capital-loss’ feature was more or less distributed evenly (excluding the zeroes), but log transformation still improved our models’ accuracies. We believe that this is due to the transformation revealing a more pronounced gap between those who had never lost money and those who had tried investing and lost money. This perhaps reflects the correlation between risk-taking and wage, or a correlation between wealthy financial background and wage (this logic may also apply to capital-gain feature). Unfortunately, we lack the resources to investigate further on this matter. Zero values were included in the following histograms.

3. **Creating dummy variables:** All the categorical features were transformed into dummy variables. This means that each unique item within the feature is transformed to its own column, where the value is set to either 0 or 1 corresponding to each data point. For example, ‘race’ feature was divided into ‘Amer-Indian-Eskimo’, ‘Asian-Pac-Islander’, ‘Black’, ‘White’, and ‘Other’ features. For all the categorical features, small minorities were grouped together into one feature, since we wanted to avoid introducing sparsely populated features as much as possible; such features would not provide enough data for ML algorithms to train on, often resulting in overfitting.

4. **Undersampling:** There was an imbalance of data where the ratio of salary_under_50K to salary_over_50K was approximately 3:1. To even out the ratio, we sampled a fraction from salary_under_50K and concatenated it to salary_over_50K, forming a new balanced data set. From this data set, we extracted the features and responses for model training and validation.

Training and Validating:

The adult wage data set was divided into a training set and a validation set: the former was used in training the models, and the latter was used to determine the accuracy of the models on data never before seen by the models. Five ML models were trained and validated:

1. Naive Bayes: Naive Bayes algorithm was run 50 times to get the average training and validation scores.
2. k-Nearst Neighbours: kNN algorithm was run 10 times to get the average training and validation score. In the pipeline, we applied MinMaxScaler to even out Euclidean distances, PCA to reduce the dimension to 15 features, and KNeighborsClassifier with n_neighbors set to 8.

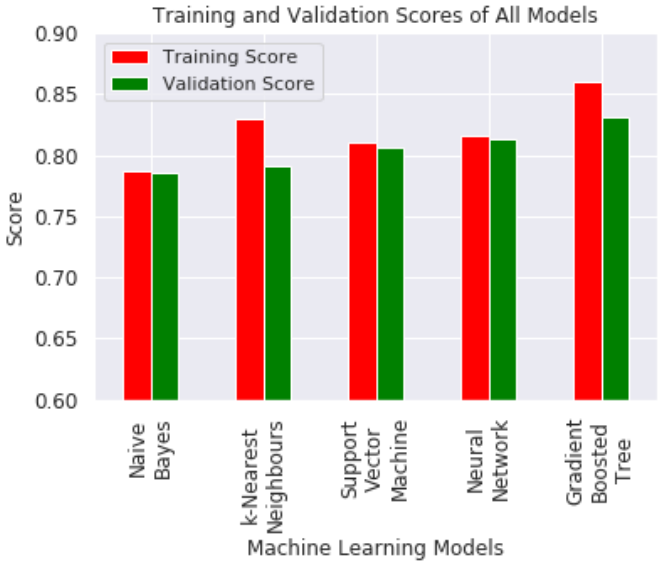
3. Support Vector Machine: SVM algorithm was run twice due to performance concerns. In the pipeline, we applied PCA(10) to speed up training without losing too much accuracy, and SVC(kernel='rbf', C=6.0, gamma='scale', decision_function_shape='ovr') to optimize the results.
4. Neural Network: MLPClassifier with two hidden layers (35 and 25 in size) was run 6 times. We set solver to 'lbfgs' and activation to 'logistic'.
5. Gradient Boosted Tree: We used 30 estimators with a max depth of 15 with 30 minimum number of samples. This algorithm was run five times to get the average scores.

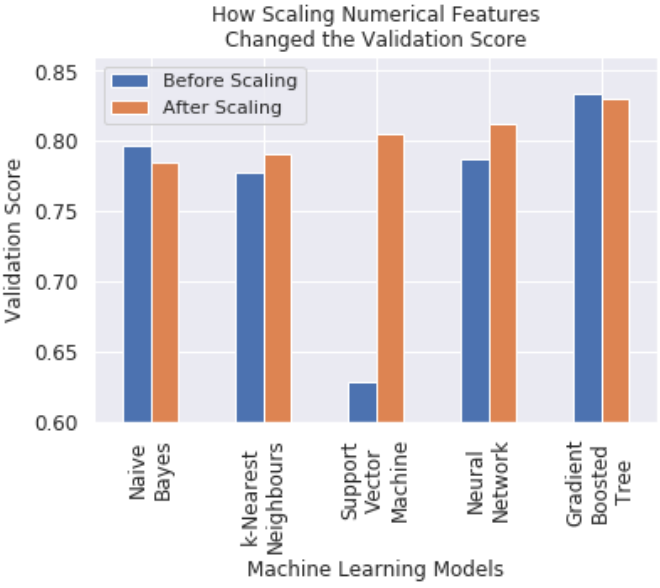
Results:

We found that Naive Bayes model yielded the lowest score at around 0.78, and Gradient Boosted Tree model yielded the highest validation score at around 0.83 on a balanced data set. The results are summarized in the bar graph to the right.

The Gradient Boosted Tree model consistently produced the highest validation score with a variety of changes made to the features, although the model suffered from large overfitting in comparison to other models. However, we believe the extent of overfitting by this model (difference of 0.03 between training and validation score) is within an acceptable range for practical purposes. Hence, we accept this model as the optimal model.

We also wanted to know whether our scaling techniques used on the numerical features had any effect on the models' scores. The results from raw data versus scaled data are outlined in the bar graph to the right. The validation score of Support Vector Machine model improved greatly after scaling; Neural Network model improved slightly; the rest of the models were not significantly affected. These findings are surprising - we expected k-Nearest Neighbours model to improve the most with scaling due to scaling affecting its calculation of Euclidean distances, but this was not the case. Since Support Vector Machines also draw





boundaries within the Euclidean space, it makes sense that scaling would affect its score. The intricacies behind why our Support Vector Machine model improved so greatly is beyond the scope of our project. We are satisfied in knowing that scaling did result in some improvements in validation scores.

Limitations:

- Undersampling removed a lot of rows. We could try oversampling with SMOTE technique.
- Combine age data with historical workforce proportion of each sector ("work class") to see if working in larger sectors is correlated to higher wage. This would take way too much research. Those data are probably not available online for free.
- We spent a significant amount of time cleaning up and analyzing other datasets (movie-wikidata option on the project requirement page), but we were not able to get a high validation score on our models. Either we could have worked more on that dataset, or we could have started work on adult wage from the beginning.

Brief Discussion on the Movie-Wikidata Failed Attempt

Our objective here was to predict review scores based on movie information such as casts, directors and plots. We used several machine learning techniques to create and train models, including converting features into meaningful variables, CountVectorizer and TfidfTransformer from sklearn.feature_extraction.text, MultinomialNB, KNN, SVM, neural network and regression.

However, all of our models had validation scores under 0.70, where most were close to 0.50. There are some possible reasons for this:

1. Many rows in the dataset had empty values, and after taking out all the NaNs, the data size decreased significantly. We could have used imputation techniques to salvage some rows.
2. When we set up ML models for "classification" (as opposed to regression), we used the method of rounding review scores then converting them to string type. As opposed to flooring or ceiling, rounding hurt our prediction because 0 to 0.5 got mapped to 0, and 4.5 to 5 got mapped to 5, while all other numbers were mapped from a range of 1. This resulted in uneven categories.
3. Casts and directors feature were transformed more practical features. For casts, we counted how many times each cast appeared in the data set to determine their popularity/experience. Then for each cast in a list of casts for each movie, we counted how many casts were highly popular (appeared in over 15 movies), moderately popular (appeared in 8 to 14 movies), somewhat popular, and etc. Directors feature was

transformed in the same way. The setting of 'casts_over15', 'casts_8to14', 'casts_1to8', 'dirs_over5', 'dirs_3to4', 'dirs_1to2' seemed to most evenly spread out the counts, and thus yielded the best prediction results.

4. Similar to our validation scores of other models, linear regression produced a poor r-value of 0.248 and r^2 score of 0.062.

Due to these difficulties we faced, we decided to stop working on this dataset and move onto something else (adult wage).

Project Experience Summary

Sean:

- Used Python's Pandas, Numpy, Scikit-learn, matplotlib, seaborn libraries to clean, process, visualize data, and make ML models. Acquired proficiency in using these libraries and basic data science skills as a result.
- Tried creative transformations of the casts and directors features of rotten tomatoes. For example, for each movie, its list of casts was transformed into counts of extremely popular casts, very popular casts, moderately popular casts, and etc. This raised the validation score of various ML models by about 5% (from 24%).
- For the movie data, I investigated the accuracy of machine learning models by visualizing the response variable using a pie chart. This revealed large imbalances, which was invalidating the accuracy of our ML models. As a result, we realized the flaw in our models and changed our direction to create more valid models.
- Reasoned and applied transformations to numerical features in adult wage data set. For example, the sample consisted of more young adults, skewing the distribution of the "age" feature to the left. I log transformed the "age" feature to generate a normal curve, making Euclidean distance calculations easier for some ML models. This had varying effects on the models we used, but most model scores increased by a small margin.
- Created optimal machine learning pipelines to quickly train and test models while maximizing their scores. Automated the summarization of results for quick comparisons and fast workflow. This allowed testing of many different transformations and combinations of features, helping us to quickly find the best settings.
- Bar graphed and wrote the results, providing a concise summary of our findings. The report effectively communicates our findings and follows up on our objectives, providing a satisfactory closure to the project.

June:
Apply machine learning techniques to clean up and analyze data to answer/predict given questions and present the result to the target audience

- Clean up, join and load movie data into single useful data file
- Apply Pipeline, CountVectorizer, TfidfTransformer, StandardScaler, MinMaxScaler, LinearRegression, MultinomialNB, KNeighborsClassifier, SVC and MLPClassifier to train machine learning models and evaluate prediction score with separate set of validation data
- Present effectively to audience by visualizing and plotting data with matplotlib.pyplot, and writing summary report that is easy to understand
- Communicate with team members on a regular basis to clarify objectives, assigned tasks, both in person and online with tools such as Slack and Google Docs
- Utilize Git by manipulating commands such as commit, push, pull, checkout, branch, clone, log, merge and tag
- Utilize official documentation for python, numpy, pandas, scikit-learn and answers from stack overflow to solve problems efficiently
- Create command line program in python with user friendly usage messages in Vim
- Write README.md which contains detailed instructions on setup and usage of the data analysis, including proper acknowledgments
- Document and explain clearly and concisely about code implementation