MIE1624H
Introduction to Data Science and Analytics

Assignment 2

Zhi Xin Zhu
1002117112
05/03/2021

1. Data Cleaning

The dataset, '*clean_kaggle_data_2020.csv',* used in this assignment is preprocessed through the file '*KaggleSalary_DataSet.ipynb*'. Data cleaning is an important process of preparing data for analysis by removing or modifying data that are incomplete and incorrect. As several values in the Kaggle dataset are missing and some of the features are categorical, data cleaning is necessary before any analysis. The data contains 10729 rows and 357 columns.

1.1 Handling Missing Values

The first procedure of handling missing values is to remove columns with at least 30% of NaN entries. For the purpose of EDA and implementation, at least 5000 training samples are required, Since we are doing a 70:30 train test split, we need at least 7142 samples in total (67% of original data).To maintain enough samples for further model building, we need need to remove features that have more than 30% of missing values. After this procedure, there are 20 columns remaining. Columns related to Q7, Q9-Q10, Q12, Q14, Q16-Q19, Q23, Q26-Q37, Q39 are dropped. There is one column of Q7 left. This column represents whether the respondent uses python. The majority of the columns dropped are from the second half of the questionnaire. One reason is most of these questions are multiple choices (select-all if applied). Removing features with a low response rate helps reduce the time required for training the regression model, but it may also lower the training accuracy and prediction accuracy since much information is eliminated.
The second procedure of handling missing values is removing columns that are irrelevant or modify or remove columns that have NaN values. Detailed steps of the second procedure are listed below.:

1) Remove the first column and Q24. The first column simply represents the time it took for the respondent to complete the questionnaire and it is irrelevant for our analysis. Q24 is encoded and bucketed and store in other columns. For the purpose of our analysis, we will be working with the encoded version of this column, therefore we don't need this.
2) Q7_Part_1: One of the choices of a multiple-choice question: Which Programming language do you use on regular basis? This is column is Python, any entry with value in this column represents that the respondent uses Python. We will modify this column into a binary question: Do you use Python on regular basis? 1 means the respondent uses Python and 0 otherwise.
3) Q38 has 1253 NaN entries, replace NaN with 'Unknown'. Q38 contains more than 10% missing entries, replacing NaN with 'Unknown' helps to preserve these entries and a certain degree of information provided by the feature
4) Q8, Q11, Q13, Q15, have the missing values for the same rows, all 561 rows are dropped. The amount of data dropped in this step is around 5%, it should not have a
5) Q25 has 159 missing values. That is around 1% of our dataset. It won't have too much impact if they are removed.

After these two procedures, there are in total 10034 entries and 18 columns left. The column names are then renamed based on the original questionnaire.

1.2 Encoding Categorical Data

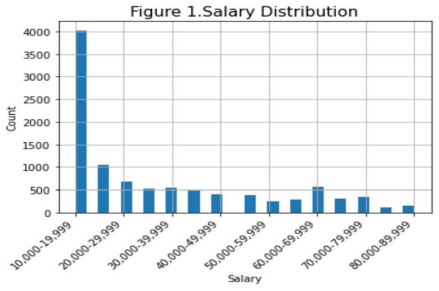
Most of the data in this dataset are categorical data. These data must be converted into numerical data so that it can be used as inputs from the prediction algorithm. One hot encoding algorithm is one of the common encoding methods used to encode categorical data. This algorithm converts each class into one binary column with 0,1 indicators. This method avoids the assumption that there are ordered relationships between categorical data, therefore it improves model performance by avoiding overfitting. However, the lack of ordered relationship property makes it a poor algorithm for encoding this dataset because some categorical data in this dataset has an ordered relationship. In addition, one hot encoding results in many lengthy columns since ut creates a column for each unique categorical input.
Instead, manual encoding and the *sklearn.preprocessiong.LabelEncoder()* function is used to encode the data. All categorical data with orderly relationships will be manually encoded. The rest will be encoded with *LabelEncoder().* The ordered relation in these data must be conserved since we are doing an ordinal logistic regression analysis. The gender column is also manually labelled simply for interpretation convenience. The Salary_Buckets column will be removed since we will only need encoded data for analysis. As a result, the processed dataset has 16 encoded feature columns and 1 encoded target column.

2. Exploratory Data Analysis

2.1 Trend and Distribution in Data

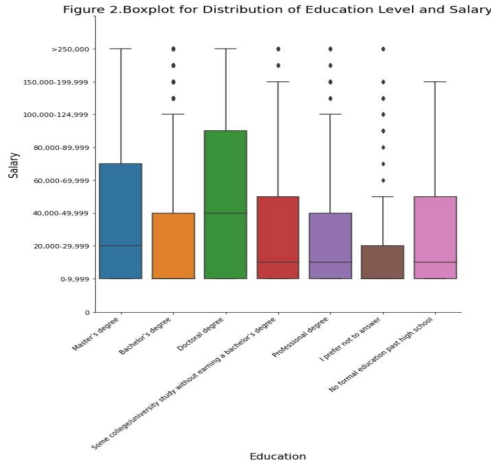
Analyzing the trends in data can help with the task of predicting the target and understanding the data. To explore the relationship between features, 3 distribution figures are constructed.

1) Histogram of Salary Distribution



From this histogram, it can be seen that the salary class (0-9,999) USD is the most distributed. The distribution of the salary is right-skewed.

2) Boxplot for Distribution of Education Level and Salary.

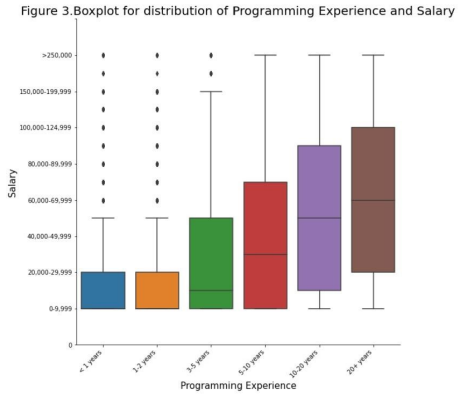


From figure 2, it can be seen that the education level of individuals does affect the salary. In general, individuals with higher education levels have higher salaries. Among all, individuals with Doctoral degrees have the highest median and 75 percentile salary. Individuals who preferred not to answer rarely earned over $50,000.

3) Boxplot for Distribution of Programming Experience and Salary
From figure 3, it can be observed that programming experience has a significant effect on the salary.
As the coding experience increases, the salary increases as well. This suggests that the experience has a strong

positive correlation with the salary earned. This boxplot also confirms that this feature has an orderly relationship among different classes.

Figure 3.Boxplot for distribution of Programming Experience and Salary



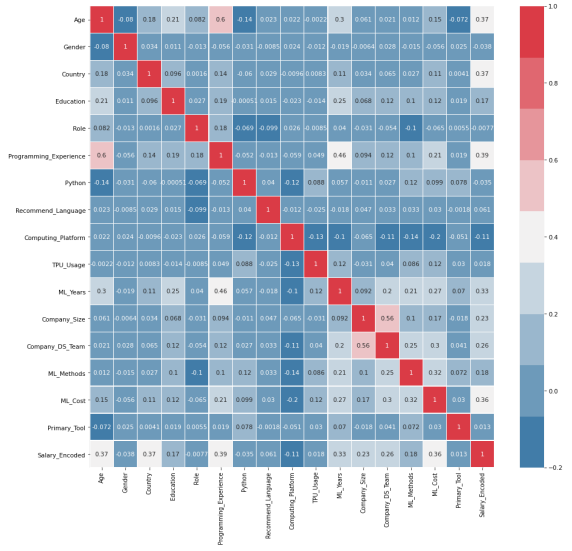
2.2 Visualize the Order of Feature Importance

To visualize the order of feature importance, a Pearson Correlation heatmap can be constructed using the seaborn library. A Pearson correlation coefficient is between -1 and 1 indicates the extent to which two variables are linearly related. The negative coefficient represents a negative correlation. A higher magnitude of number (large |coeff|) represents a stronger correlation. From the heatmap, we observed the following.

Table 1. Most Correlated and Least Correlated Feature with Salary

Most Correlated	Least Correlated
Programming Experience (p = 0.39)	Role (p = -0.0077)
Age (p = 0.37)	Primary Tool (p = 0.013)
Country (p= 0.37)	TPU Usage (p = 0.018)
ML_Cost (p = 0.36)	Python (p = -0.035)
ML_Years (p = 0.33)	Gender (p = -0.038)

Figure 4. Pearson Correlation Heatmap



3. Feature Selection

Feature engineering is a process of using domain knowledge of the data to create features that make machine learning algorithms work. If feature engineering is done properly, it will increase the predictive power of machine learning algorithms by creating raw data that help facilitate the machine learning process. However, if feature engineering fails to translate the crucial information in the data, it will aggravate the performance of the model. In this section, different feature engineering and selection algorithms are applied to explore the importance of each feature and its correlation with the target variable. Feature selection not only speeds up the algorithm but also eliminates some unnecessary noise of the data.

3.1 Lasso Regression (L1 Regularized Regression) Method

Regularization is a technique used to reduce the freedom of the model by adding penalties to different parameters of the model. For regularized regression, the penalty is applied over the coefficients that multiply each of the predictors. The Lasso (L1) regularized regression has the property that is able to shrink some of the coefficients to zero such that certain features are removed from the model, therefore this is a suitable algorithm for feature selection. A threshold can also be added. If the feature has a non-zero but smaller than the threshold coefficient, it will also be removed from the model due to an insignificant relationship with the output. Two different thresholds are applied, they are 0.05 and 0.5. With a small threshold of 0.05, no features are eliminated which suggests that all features contain some extent of information for the prediction. With a higher threshold of 0.5, 5 features are removed. These features are 1) TPU_Usage; 2) Primary_Tool; 3) Country; 4) Programming_Experience; 5) ML_Years.

Compared with the Pearson correlation heatmap from 2.2, only two removed features matched; TPU_Usage and Primary_Tool. The other three removed features are actually 3 of the most correlated features in 2.2. This is normal because the mathematical algorithm used is different. We will also use other algorithms to perform feature selection and choose the features that show up most frequently among all algorithms.

3.2 Chi-Square Test

A chi-square test can be used to test the independence of two variables. A large chi-square value suggests that the feature is more dependent on the target whereas a small chi-square value suggests independence. The five most independent features that should be removed are 1) Python; 2) Primary_Tool; 3) Recommended_Language; 4) Role; 5) Gender.

Compare with Pearson correlation, most features matched except for "Recommended_Language". Compared with Lasso regression, only the 'Primary_Tool' feature matched.

3.3 Recursive Feature Elimination (RFE)

RFE is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached. Features are ranked by the coef_ or feature_importances_ model attributes and by recursively eliminating a small amount of feature per loop. The RFE method attempts to eliminate dependencies and collinearity that may exist in the model. As we want to eliminate the worst 5 features, the RFE parameter should be 11 (16-5). The features that are removed by this algorithm are: 1)TPU_Usage; 2) Primary_Tool; 3) Gender; 4) Computing_Platform; 5) Python.

Compared with Pearson correlation, most features matched except "Computing_Platform". Compared with Lasso regression, only TPU_Usage and Primary_Tool matched. Compared with the chi-square test, all matched except for Computing_Platform.

3.4 Feature Selection

We want to remove the 5 least correlated features from our dataset. Combining the result of all analyses and the correlation plot, we want to pick the features that showed up the most among all algorithms. The following five features are selected: 1) TPU_Usage; 2) Primary_Tool; 3) Gender; 4) Role; 5) Python.

Encoding non-ordered data into ordered data will confuse any algorithms. To prevent this problem, selected features with no obvious ordered relationship between inputs will be encoded using the One-hot encoding with get_dummies() function. The final dataset has in total 10034 entries with 87 columns including the target variable.

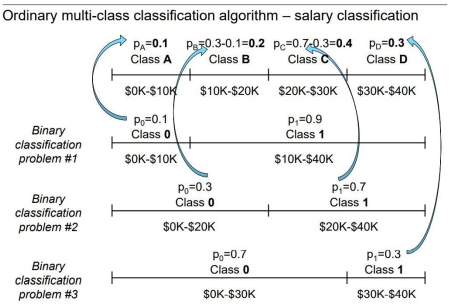
4. Model Implementation and Hyperparameter Tuning

4.1 Ordinal Logistic Regression

Model Logic: The ordinal logistic regression model implemented in this section is modified from the *sklearn.LinearModel.LogisticRegression()* based on the 'Ordinary Multi-class Classification algorithm' diagram shown in Figure x. The concept is quite similar to the OVR algorithm. The basic idea is breaking down a multi-class problem into multiple binary

classification problems. If we have *n* classes (A, B, D, etc) for our target variable, we define an *n*-1 binary classification problem that calculates the probability of a data point in Class 0 or Class 1, where for problem *i*, the probability of Class 0 represents the probability of a data point in Class A to Class *i* and the probability of Class 1 represents the probability of a data point in Class *i+1* to Class *n*.

Figure . 5 Ordinary Multi-Class Classification



The data has in total 15 salary buckets, therefore we need to define 14 binary logistic models. Since we are training 14 models, the target variable must be encoded differently for each binary model. For example, for the first model mm0, Class 0 is in Class 0 and all other classes are considered as Class 1. For mm1, Class 0 and Class 1 are considered Class 0 and all other classes are considered Class 1 and so on. The encoded targets are stored as y0, y1, and etc. Then the models are trained using 10 fold cross-validation. Instead of making a prediction directly using *model.prediction()*, we use *model.predict_proba()* to retrieve the probability of a data point belong to which class. We then subtract the probability of these models to get the exact probability for all 15 classes. For each data point, the sum of probability for all 15 classes should equal 1. The prediction is made based on the highest probability.

The model training performance is shown below.

```
Fold 1 Accuracy: 40.26 %
Fold 2 Accuracy: 41.54 %
Fold 3 Accuracy: 39.26 %
Fold 4 Accuracy: 39.74 %
Fold 5 Accuracy: 38.75 %
Fold 6 Accuracy: 37.61 %
Fold 7 Accuracy: 43.30 %   Average Accuracy: 40.18 %
Fold 8 Accuracy: 42.59 %   Performance Variance: 2.87
Fold 9 Accuracy: 39.03 %   Average Variance: 0.09
Fold 10 Accuracy: 39.74 %  Average Bias: 28.62
```

The training accuracy is around 40.18%. The variance of the prediction is very low but bias is high.

4.2 Hyperparameter Tuning

The LogisticRegression() function has numerous parameters, they are*:*penalty=, dual, tol, C, fit_intercept, intercept_scaling, class_weight, random_state, solver, max_iter, multi_class, verbose, warm_start, n_jobs, l1_ratio*

We will choose 'C' and 'solver' for this hyperparameter tuning analysis.

C: positive float value; it is the inverse of regularization strength. smaller value represent stronger regularization. The default value is 1.0. For tuning we will try C = [0.01, 0.05, 0.1, 0.5, 1.0, 5.0, 10.0, 50.0]

Solver: Algorithm to use in the optimization problem. Algorithm includes [*'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'*]. default = *'lbfgs'*

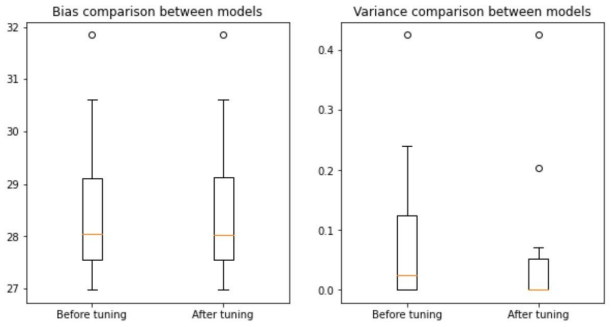
The grid search method is used for tuning the hyperparameters. The optimal C is 0.01 and the optimal solver is 'newton-cg' and it yields training accuracy of 40.21%, bias of 28.61, and average variance of 0.0701.

4.3 Model Selection based on Bias-Variance Trade-off

In machine learning, one of the performance metrics to evaluate a model is test (generalization) error. The generalization error is the error in unseen data which can be decomposed into bias error, variance, and irreducible error. In order to reduce the bias, the model complexity should be increased. However, an over-complex model will result in high variance. This is called bias-variance trade-off. For optimal algorithm performance, the bias and variance should both be as low as possible.

Out of the two models, the tuned model has higher accuracy. From the perspective of low bias and low variance, the tuned model has a slightly lower bias and lower variance thus it is a better model. The comparison can be seen in figure x.

Figure 6. Bias and Variance Comparison



5. Test and Discussion

The optimal model using the set of parameters (C= 0.01, solver = 'newton-cg') yields a training accuracy of 40.21% and a test accuracy of 41.98%. The optimal model is generally underfitting because, from the distribution of true target and prediction, it can be seen that most of the predictions made are the salary bucket of 0 (0-9,999), 10 (100,000-124,999), and 12 (150,000-199,999). Only a few predictions were correct for other salary buckets which suggests that the model may have oversimplifying assumptions about the dataset.

Improvements:

- To avoid underfit model, the number of features should be increased thus the hypothesis space can be expanded.
- Improve the quality of the original dataset. During the data cleaning stage, most of the original feature columns were dropped due to an extremely low response rate. Some of the feature columns' inputs were replaced with a new categorical entry 'Unknown' or the mode, which harms the quality and make the dataset more difficult to be differentiated by the regression algorithm
- Improve the label/encoding procedure for the categorical inputs. As some columns have been encoded manually to emphasize the obvious ordered relationship, other columns that contain non-obvious relationships are not recognized. Take 'Country' for example, one hot encoding was used to treat each country equally as feature inputs. However, some developed countries are inherently better than other countries in the aspect of salary earned. Therefore, orderly encode the categorical inputs with not so obvious ordered relationship should improve the model performance.
- Improve the process of feature selection and engineering to add enough model complexity. Simply select the most important features may not be sufficient to develop a machine learning model with appropriate complexity. Apply feature engineering to the existing features helps expand the hypothesis space by adding new features based on existing features.