MIE1624 Assignment 2 Report

Kaiyan Jiang 1003848189

March 11, 2022

The assignment aims to train, validate, and tune multi-class ordinary classification models that can determine the 2021 Kaggle's survey respondent's yearly salary bucket. Based on the column 'Q25' "What is your current yearly compensation (approximate $USD)?", the target column 'Q25_buckets' has been obtained by combining some salary
buckets in the column 'Q25'. And column 'Q25_Encoded' has been obtained by label encoding the column 'Q25_buckets'.

1. Data Cleaning

First, remove useless features and the question row from the dataset. There are many multiple choices questions in the survey, and the option "OTHER" seems not have a meaning. The participants also did not specify it, so remove all columns contains option "OTHER". And remove the features that are no longer of useful since they don't show anything discernable, or they've been aggregating into another feature. Response time is not related to target variable. 'Q25' and 'Q25_buckets' are removed because 'Q25_Encoded' can replace them.

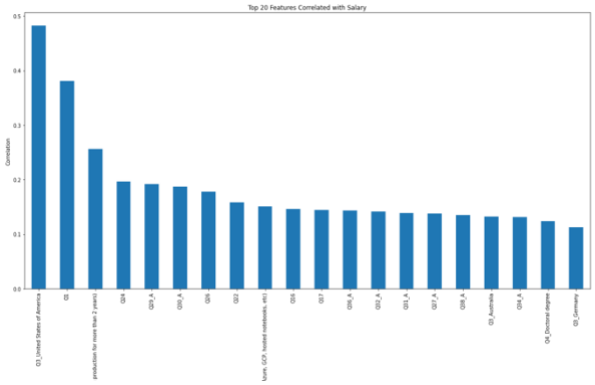
Next, encode the categorical data. Selecting all columns with categorical responses. Count the missing values. From the results, columns 'Q8','Q11','Q13','Q15','Q26','Q28','Q33','Q35','Q41' have missing values. To deal with the missing values in these columns, first print the unique elements in each column. If the questions have a response, such as "None", "Never", replace the missing value with response like this. Since the participant chose not to answer the question, it is reasonable to assume the participant do not have any experience before. For questions do not have such responses, replace the missing value with the mode as it identified the general level of all participants.

The categorical questions can be separated by the type of responses: nominal and ordinal. Encoded nominal features with *pandas.get_dummies()*, and merge them with the original dataset. Use *LabelEncoder()* from scikit-learn to convert the rest ordinal data into numerical data.

Last two steps of data cleaning are encoding multiple choices columns and reorder the dataset. Select each column for multiple choices questions and encoded non-missing value with 1 and missing value with 0. Then, combined all columns for one question into one column, thus, each column shows the sum of answers that participant chose. Next, reorder the dataset by the order of question and put the target column 'Q25_Encoded' at last.

2. Exploratory data analysis and feature selection

First a correlation plot is drawn to show feature importance (See Appendix Figure 1). The correlation plot is shown as a heatmap with range from -1 to 1. For column "Q30_B" as all its entries are 0 so it does not have correlation with other features. However, since there are too many features, the correlation plot is very redundant. A plot that identifies the top 20 features that have strong correlation with salary "Q25_Encoded" is plotted.



From the plot, the most important feature is 'Q3_United States of America', with a correlation 0.482, which means if the participant from America, the person will have a great impact with the yearly salary. 'Q1', that represents age, follows next, with a correlation 0.381. 'Q23_ We have well established ML methods','Q24' and 'Q29_A' are the top 3-5 features, which represent whether employer incorporate machine learning methods in business, activities that

make up an important part of role at work, and cloud computing products respectively.

After data cleaning, the dataset ends up with 218 features, which is a large number and may likely cause overfitting. Feature engineering should be applied on the dataset as the number of features need to be reduced to improve the performance of ordinal logistic regression algorithm by selecting the most related features. I fitted a Lasso Regression[1] on dataset and select those features that have a coefficient different from 0. However, after applied the Lasso Regression, there is only one feature that have coefficient equal to 0. Thus, I also applied PCA to reduce dimensionality. Now the number of features is 50.

3 Model Implementation

Ordinal logistic regression algorithm using 10-fold cross-validation is implemented in this section. The algorithm uses a feature dataset and a target array in fitting. The output is the probability matrix that each entry of features belongs to each of the salary buckets. The dataset is divided into training set *x_train, y_train* and test set *x_test, y_test* by *test_size* = 20%. Standardization is not performed on dataset since the dataset is already standardized before performing PCA.

```
Fold 1 |Train Accuracy: 0.46958483754512637|Validation Accuracy 0.4788961038961039
Fold 2 |Train Accuracy: 0.47129963898916966|Validation Accuracy 0.45535714285714285
Fold 3 |Train Accuracy: 0.47143759588484796|Validation Accuracy 0.46303818034118605
Fold 4 |Train Accuracy: 0.4732424871401498|Validation Accuracy 0.46791226645004064
Fold 5 |Train Accuracy: 0.4719790632614385|Validation Accuracy 0.4476035743298132
Fold 6 |Train Accuracy: 0.4722497969497338|Validation Accuracy 0.46385052802599513
Fold 7 |Train Accuracy: 0.47152784044761303|Validation Accuracy 0.4703493095044679
Fold 8 |Train Accuracy: 0.4700839274433715|Validation Accuracy 0.4646628757108042
Fold 9 |Train Accuracy: 0.4754986012092771|Validation Accuracy 0.4281072298943948
Fold 10 |Train Accuracy: 0.47279126432632435|Validation Accuracy 0.45004061738424045

Average Train Accuracy 0.4719695053197052
Average Validation  Accuracy 0.4589817828394189 3

Train Accuracy Variance 2.777522651088795e-06
Validation Accuracy Variance 0.0002059747047652797
```

When fitting model, using *KFold.split()* split the training set *x_train* into new training set and validation set for each fold. This is the model accuracy on training set and validation set on each fold. The average accuracy on training set is 47.196% with a variance 2.7775e-06. The average accuracy on validation set is 45.898% with a variance 0.0002059. In the next few steps, parameters ‘C’, ‘solver’ and ‘penalty’ are selected and tested. First, ‘C’ is the inverse of regularization, the smaller ‘C’ value means a stronger regularization. A list of ‘C’ values [0.0001,0.0005,0.001,0.005,0.01,0.05,0.1,0.5,1] are used in model, and the bias-variance trade-off on validation set performance is calculated and plotted. From the graph, the bias decreases and the variance increases along with the increase in ‘C’ values. When ‘C’ is greater than 0.005, the extent of changes is very small. From the bias-variance trade-off, models perform better as the value of ‘C’ increases. And the accuracy of predictions also increases with ‘C’ increases.

‘Solver’ is the Algorithm to use in the optimization problem. Different ‘solver’ algorithms (‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’) are tested. The bias-variance trade-off is also calculated and plotted. From the graph, there is no difference when using ‘newton-cg’, ‘lbfgs’ and ‘sag’. At last, ‘penalty’ has been tuned. From the description of model, the ‘penalty’ can be one of the following: ‘l1’, ‘l2’, ‘elasticnet’ and ‘none’. Since the default ‘solver’ is ‘lbfgs’, only ‘l2’ and ‘none’ can be applied. After modelling, the bias-variance trade-off showed that these two kinds of ‘penalty’ have very similar performance. But ‘none’ has a slightly lower total error. Thus, the model with ‘penalty’ equals to ‘none’ performs better.

4 Model tuning

The ordinal logistic regression contains following hyperparameters: ‘penalty’, ’dual’, ‘tol’, ‘C’, ‘fit_intercept’, ‘intercept_scaling’, ‘class_weight’, ‘random_state’, ‘solver’, ‘max_iter’, ‘multi_class’, ‘verbose’, ‘warm_start’, ‘n_jobs’, ‘l1_ratio’.

From the last section, chaning hyperparameter ‘C’ and ‘solver’ have a greater impact on the total error of the model than changing the hyperparameter ‘penalty’. Selected ‘C’ and ‘solver’ two hyperparameters for model tuning. Using grid search to find an optimal model from list of ‘C’ values: [0.0001,0.0005,0.001,0.005,0.01,0.05,0.1,0.5,1] and list of ‘solver’ [‘newton-cg’, ‘lbfgs’, ‘liblinear’, ‘sag’, ‘saga’].

[1] https://towardsdatascience.com/feature-selection-in-machine-learning-using-lasso-regression-7809c7c2771a

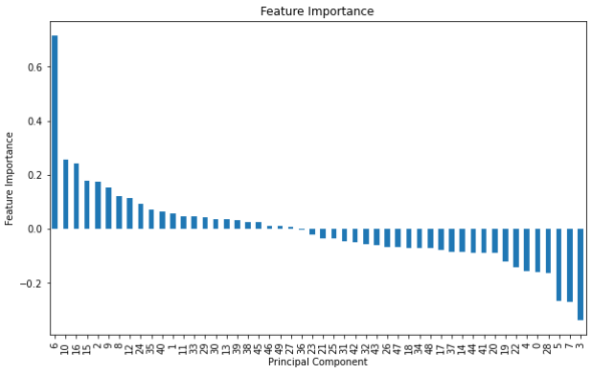
Using *GridSearchCV()* to try each combination of 'C' and 'solver' from the dictionary and evaluated the

```
Best parameters: {'C': 0.005, 'solver': 'newton-cg'}
Best Accuracy Score: 0.46824236517218976
```

model using Cross-validation method. Accuracy and weighted F1-score will be monitored through evaluation, the model with highest accuracy will be selected as the optimal model.

From the results, the best parameters are 'C' = 0.005 and 'solver' = 'newton-cg'. The highest accuracy is 46.824%.

The feature importance here is identified with coefficient. The plot shows that principal component 6 has the highest coefficient, which means that principal component 6 is the feature with strongest correlation with salary buckets.



In Section 2, the most strong-correlated feature is participants' country is from USA. Since the PCA are carried to reduce dimensionality in Section 2, the feature importance graph is showing the importance of principal components rather than the original features.

5 Testing & Discussion

Using KFold() to split the training and test sets with into 5 consecutive folds, which is equal to the test_size = 20%. Train the optimal model on training set and make classification on both training and test set.

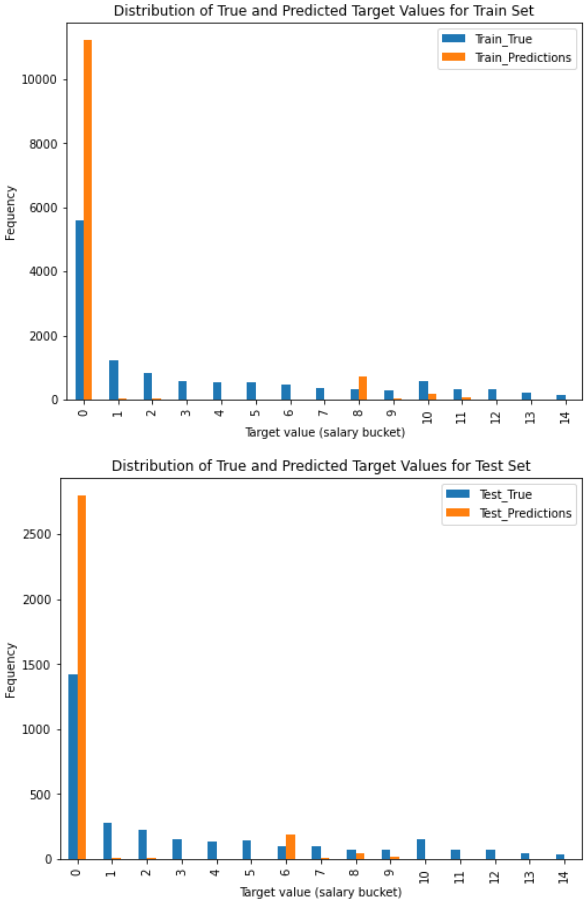
```
Fold 1 |Train Accuracy: 0.46085120207927227|Test Accuracy 0.4667099707697304
Fold 2 |Train Accuracy: 0.46154470884431087|Test Accuracy 0.46523716699155293
Fold 3 |Train Accuracy: 0.46300657841305937|Test Accuracy 0.45938921377517866
Fold 4 |Train Accuracy: 0.4677982619995127|Test Accuracy 0.4441195581546459
Fold 5 |Train Accuracy: 0.4622756436286851|Test Accuracy 0.46263807667316437

Average Train Accuracy 0.46309527899296804
Average Test  Accuracy 0.45961879727285443

Train Accuracy Variance 7.5594670158990294e-06
Test Accuracy Variance 8.2811497445433e-05
```

The average accuracy on training set is 46.309% and the average accuracy on testing set is 45.961%. The accuracies are very close, while the training set is slightly higher than test set. The accuracy of the model can be further improved by reduce to lower dimensionality or tuning more hyperparameters.

To identify whether the optimal model is overfitting or underfitting, I plotted the Mean Absolute Error (MAE) for each fold. For the most folds, the MAE of test set are slightly higher than MAE of training set. Thus, the optimal model is slightly overfitting.



According to the distribution of true target variable values and their predictions, the true value's distribution is right skewed as a lot of participants are in salary bucket 0. The figures showed that the number of predictions in bucket 0 and bucket 8, exceed the true values for training. For test data, the number of predictions in bucket 0 and bucket 6, exceed the true values. Thus, the classification model is highly likely to distinguish the participant into very low salary bucket such as bucket 0 or to some middle level, such as bucket 8 or bucket 6. And for the rest buckets, the classification model cannot predict well.

Appendix:

Figure 1: Correlation Plot Between Features

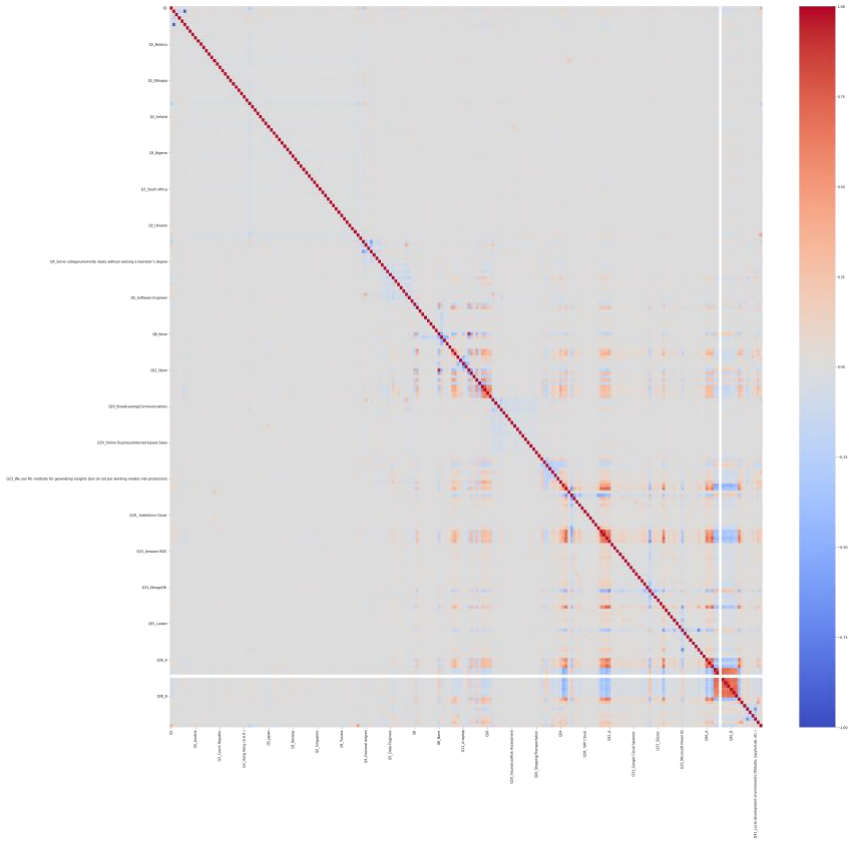


Table 1: The Top 20 Features Correlated with Salary Buckets

Q3_United States of America	0.482180
Q1	0.381054
Q23_We have well established ML methods (i.e., models in production for more than 2 years)	0.256236
Q24	0.196534
Q29_A	0.191749
Q30_A	0.187545
Q26	0.178180
Q22	0.158761
Q11_A cloud computing platform (AWS, Azure, GCP, hosted notebooks, etc)	0.150570
Q16	0.146733
Q17	0.144080
Q36_A	0.143555
Q32_A	0.141483
Q31_A	0.138422
Q27_A	0.137807
Q38_A	0.134901
Q3_Australia	0.132294
Q34_A	0.131066
Q4_Doctoral degree	0.124057
Q3_Germany	0.112712

Name: Q25_Encoded, dtype: float64

Figure 2: Bias-Variance Trade-Off for C

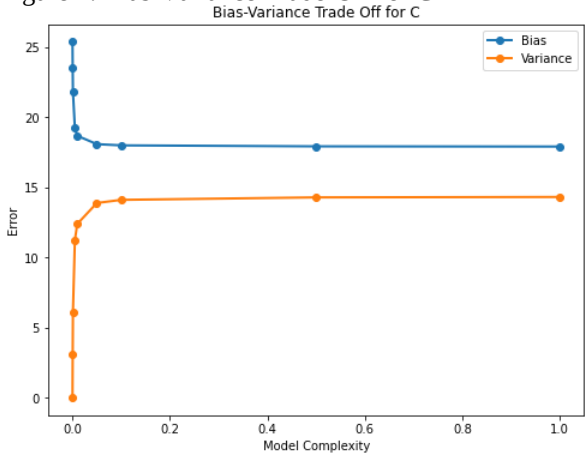


Figure 3: Bias-Variance Trade-Off for Solver

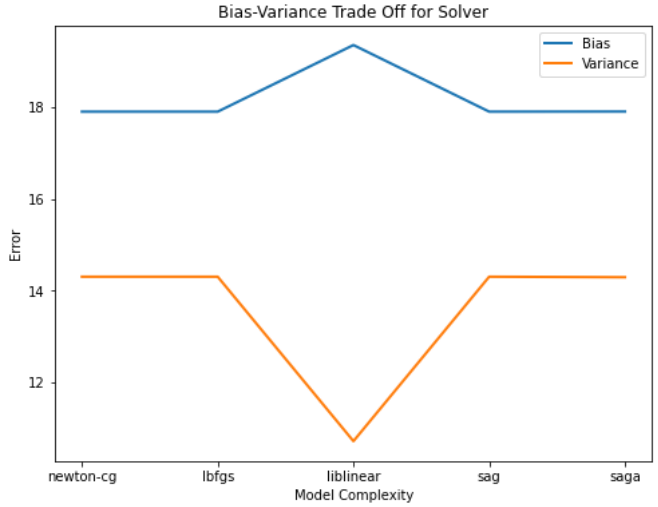


Figure 4: Bias-Variance Trade-Off for Penalty

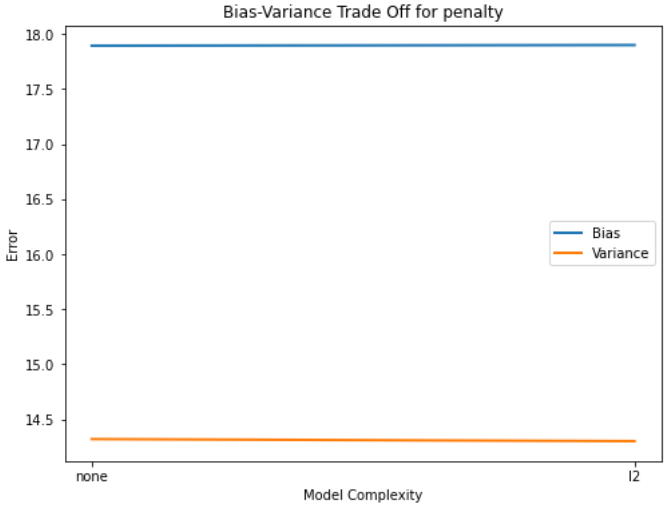


Table 2: Classification Report on Test Data

	precision	recall	f1-score	support
0.0	0.53	0.96	0.68	1423
1.0	0.17	0.01	0.03	281
2.0	0.04	0.00	0.01	225
3.0	0.07	0.01	0.01	153
4.0	0.04	0.01	0.01	132
5.0	0.12	0.02	0.04	140
6.0	0.04	0.01	0.02	95
7.0	0.00	0.00	0.00	103
8.0	0.25	0.01	0.03	69
9.0	0.00	0.00	0.00	74
10.0	0.22	0.30	0.25	151
11.0	0.13	0.04	0.06	70
12.0	0.22	0.30	0.25	74
13.0	0.12	0.02	0.04	49
14.0	0.00	0.00	0.00	40
accuracy			0.47	3079
macro avg	0.13	0.11	0.09	3079
weighted avg	0.30	0.47	0.34	3079

Figure 5: Mean Absolute Error of Each Fold