ASSIGNMENT 2 REPORT

PART1. Data cleaning

There are 2 situations to consider before dropping the columns with NANs.

First, the column description contains a substring which says '-Selected Choice-'. The NAN values in this kind of columns means the participant does not select this choice and we simply treat it as a '0'. We cannot drop it otherwise we may lose useful features.

Second, NANs exists in the column and the column description does not contain '-Selected Choice-'. There are like 10 columns(survey questions) that has NAN values(except the first case mentioned above). I choose to drop them because it is not a good idea to fill the missing values especially for survey data. For numeric values, we can use techniques like imputation to approximate but for our survey data but most of them are strings/multiple choices and it is not meaningful to approximate. Even if some questions' answer choices are numeric, approximating it impact our model negatively because the 'data' is not real. The approximation may harm the model performance later as well. I want to preserve the reality of the data. Also, we only have around 10 columns with NANs that are dropped, most of the feature data are preserved and transformed for our use later. The dropping may not impact much.(we have around 450 columns after data cleaning)

(We can see that some column description contain '-Selected Choice'(one dash less). These questions require choosing one choice from several answer choices. If NAN exists, we drop the column as well. This is included in the second case mentioned above)

Survey data values are missing maybe because

1.The participants do not want to or forget to answer the questions.

2. The participant cannot find any answer choices that best describe his/her situation.

I use one-hot encoding instead of label encoding/ordinal encoding here.

Label encoding is suitable for nominal data. While label encoding is suitable for ordinal data. Nominal variable comprises a finite set of discrete values with no relationship between values while ordinal variable comprises a finite set of discrete values with a ranked ordering between values.

For our dataset, our features are the survey questions and we do not have the ranked order between values. (maybe we can say Q4 has a ranked order of degrees but most of the columns do not have this ranked order between values) The most important reason I give up using label encoding is that the label numbers may mislead the model, making the model think that there is a ranked order between instances. This is not the case here. On the other hand, One-Hot Encoding is suitable for categorical variables. It creates additional features based on the number of unique values in the categorical feature. Every unique value in the category will be added as a feature. The binary values makes our data more useful and expressive and makes the problem easier to model.

PART2. Exploratory data analysis and feature selection

Top ten features that are highly correlated to participants' yearly compensation:

I rank the correlation values for the features so the above ten features are the features that are most correlated to participants' yearly compensation.

Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the models. Sometimes it create new artificial features that are not in the training set. The intention is to improved model accuracy on unseen data. Apply feature engineering is very important for our task here. Actually one-hot encoding we just applied is a technique of feature engineering. Techniques like scaling, ouliers handing and imputations are also included in feature engineering. We have 450 features and some of them are highly correlated features. We may need to remove one for each pair of highly correlated features. The data has high dimensions and we need to apply dimensionality reduction techniques. Otherwise we may have the curse of dimensionality issue.

We may want to extract the features that are most related to the target for better prediction accuracy. We also have imbalanced dataset here which we may apply some techniques like oversampling to make our data more balanced. Otherwise the model may not learn to classify properly. Because not all features are important for predicting the target. we will apply 2 feature engineering techniques below to do dimensionality reduction.

First, we use ExtraTreesClassifier to get the feature importance of our dataset

ExtraTreesClassifier(Extremely Randomized Trees) is a method based on decision trees. It resembles random forest and randomizes certain decisions and subsets of data to prevent over-learning from data and overfitting. We can directly do feature selection based on its feature_importances_ method and choose the amount of features we want to keep. Simply dropping columns based on feature_importances_ method of ExtraTreesClassifier is not good enough. We want to apply pca to further reduce dimension while preserving the variance of data. Another reason is that we want our pca model to

ignore those 'less important' features and focus only on those most relevant features. So we first filter those features out to let pca capture the real essence of the data.

Next, we want to apply pca for further dimensionality reduction.
Principle component analysis is a tool to 'summarize' the existing data and features. In other words, it find a more meaningful basis or coordinate system for our data and find the strongest features of the samples based on some covariance matrix. The goal is to extract the important information from the data and to express the information as a set of summary features called principal components. The principle components are just like our features but they do not have names. The explained variance shows how much variance a principle component preserves. Usually, the first few principle components have majority of the explained variance. By settng n_components to 0.8, I am asking pca to give me the top principle components that explained 80% of our data variance. It is a good technique to reduce data dimension and tackle the curse of dimensionality.

PART3. Model implementation

We should apply normalization/standardization before applying pca but we may or may not apply scaling to the resulting data of pca.
After applying pca the data are more or less of the same scale(not exactly on the same scale) and we normally scale data if the range values for features are different. But we can also scale the data to keep them in a range of (0,1) so that our model does not bias towards larger values. PCA calculates a new projection of the data set. In other words, we get new features. And the new axis are based on the standard deviation of the variables. So a variable with a high standard deviation will have a higher weight for the calculation of axis. Normalizing the data before applying pca makes all variables have the same standard deviation and weight. But since we only have binary data (one-hot encoding), It is not necessary to do the scaling as well. They are already in scope(0,1).

The accuracy across folds does not vary too much. When C = 0.005, penalty = l2, solver = saga , we get our best model.
For logistic regression, the smaller the C, the stronger the regularization. l1(Lasso) has more regularization power than l2(Ridge). Regularization is a way to prevent overfitting and to tackle high variance. And We want to penalize or adjust each weight of the independent variables so that it makes a good prediction on a test set that it has not seen before. We want to choose a middle point of having high bias and high variance. C is 1/lambda. If we decrease C, we are essentially decrease the model complexity. So the variance decreases and bias increases. The more complex the model, the lower its bias and complexity will move the model to capture the data points, hence its variance will be larger. But if variance is high, the model is not going to do well on the test data because the model is more "spread out".From the result we can see that our best model has the best validation score which means it generalizes what it learns well and it achieves a good balance between bias and variance. In other words the balance between model complexity and generalization ability. We also need to be careful not to penalize the model too much, otherwise our model will be underfitted, which means the model does not perform well on both the training and test set.

PART4. Model tuning

Penalty, C and solver are the hyperparameters I use.

I would choose penalty and C because strictly speaking, solver is not a hyperparameter(also mentioned in lecture). But I include the solver option here just to see the output. I only use liblinear and saga solver here because only these 2 support both l1 and l2 regularization. C and penalty are the regularization terms for logistic regression. Determining their values are critical for getting a good prediction model for the test set. Small C indicates strong regularization and l1 has stronger regularization power than l2. We need to carefully choose their values to seek a balance between bias and variance. The purpose of regularization is to prevent overfitting so that we can get reasonable result on the test set. But we also need to be careful that when regularization is too strong, the model tends to underfit. Which has high bias and high variance. If the penalty is too weak, the model may overfit the training data. Which has low bias and high variance.

I use F1 score here because of the following reasons.

Our dataset is imbalanced, for example, we have target 0 much more than other targets(also shown below). This may create problems because the model is trained more on class 0 instances. Accuracy rate is calculated as the number of right answers/the number of total answers. So we need to introduce metrics that give more weights to the false positive and false negative rate. In many cases, our model may give good predictions for actual class 0, but bad predictions for other classes because the models learns more about class 0(imbalanced dataset).

Precision, Recall and F1 score are crucial indicators to judge a imbalanced dataset. F1 score is a comprehensive metric which take into account the precision and recall. f1 = 2*precision*recall/(precision+recall), precision = TP/(TP+FP) and recall = TP/(TP+FN). We can see that f1 score take into account the FN and FP rate. In other words, it take into accounts how our data is actually distributed. And there maybe large amount of actual negatives in our dataset. So because FN and FP is important to evaluate our model here, we use f1 score instead of accuracy.

PART 5. Testing & Discussion

The accuracy and f1 score does not vary much when comparing the performance of test set and training set.

To increase the accuracy, the following may be applied:

1.use more principle components to do the prediction. In this assignment, I only use half of the principle components(has about 80% of the total explained variance).

2.use oversampling technique(using SMOTE mentioned above) to make our dataset more balanced will likely to increase accuracy. SMOTE is an oversampling technique which is used to increase the data points for minority classes by augmenting data with similar data points as of the minority class.

3.Obviously, we can try more hyperparameters(use more slovers and try penalty = elasticnet, etc). We can also try hyperparameters like 'class_weight', ‘random_state’ for getting a model with better accuracy.

4.We use the importance from ExtraTreeClassifier as our first tool and PCA as our second tool to reduce data dimension, which can be replaced by other techniques like LDA. It is similar to PCA. LDA finds the components that maximize both the variance of the data and the separation between multiple classes. It is often used for classification models.

We can also use SVD, Contrary to PCA, it does not center the data before computing the singular value decomposition.

Also, simply taking features that are highly correlated to the target is plausible as well and worth trying.

Overfitting refers to the situation where the model performs well on training data but badly on the test data.

Underfitting refers to the situation where the model performs badly on both training and test data.

From the result we get, there is no overfitting and underfitting problem. The accuracy and f1 score for training set is around 3% higher than the test set, which is totally acceptable. And the accuracy of the test set is similar to the result we get from validation set in part 3 and 4. So it indicates that there are no severe issues of underfitting and overfitting and it seems that our model has found a good balance between variance and bias and the regularization we imposed is moderate.

The insights I gained are as follows

1.Proportion of participants and their yearly compensation:

0-9,999 : 41.4%

10000-19999 : 10.4%

20000-29999 : 6.7%

30000-39999 : 5.0%

40000-49999 : 5.1%

50000-59999 : 4.8%

60000-69999 : 3.8%

70000-79999 : 3.7%

80000-89999 : 2.5%

90000-99999 : 2.6%

100000-124999 : 5.3%

125000-149999 : 2.9%

150000-199999 : 3.2%

200000-249999 : 1.1%

>250000 : 1.4%

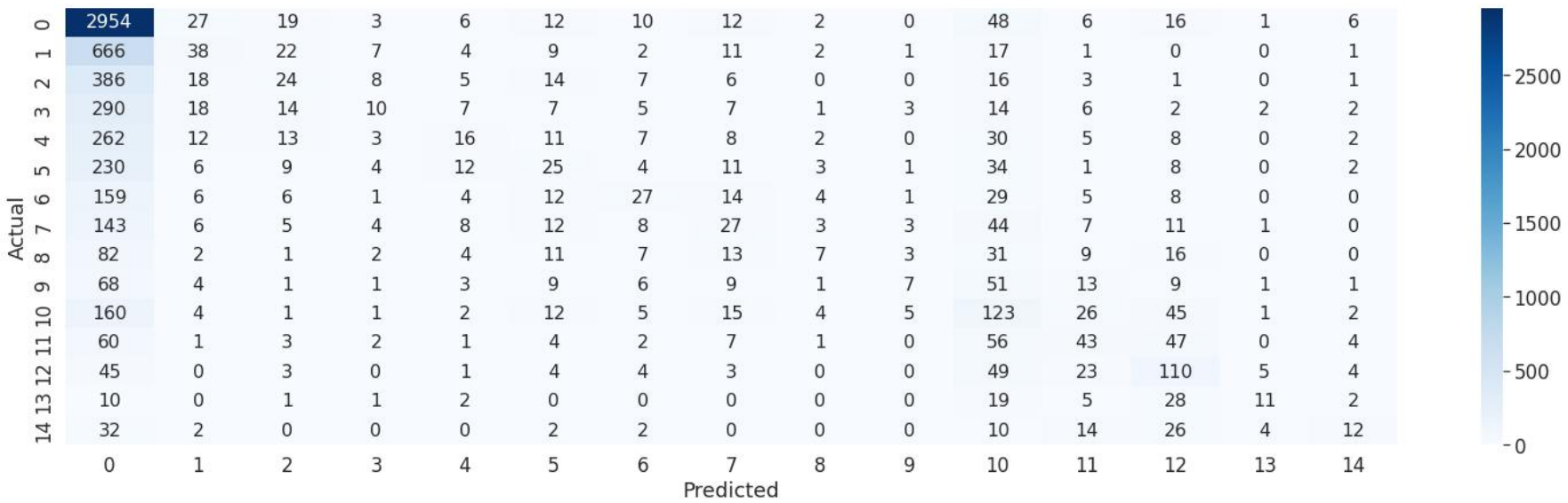
We can see that half of the participants have a yearly compensation <= 19999. The distributions of number of participants having higher yearly compensation classes are sparse.

2. From the feature_importance_ of the ExtraTreeClassifier, we can see that whether the participant is from the US or not is highly related to their yearly compensation.(almost 2 times the importance of the second important feature)

3. the recall for our best model is acceptable(result similar to accuracy score), but the precision is bad(over 10 percent lower than accuracy and recall). Which means the model has more average FP cases than FN cases. We cannot directly see this from the accuracy score, but we can use metrics like f1 score to show the combination of these 2 indicators.

The plot of confusion matrices of training set and testing set are as follows:

Training:



Testing: