Multi-Class Logistic Regression Report

Data Cleaning

With the original shape of the dataset being (10730, 356), which contains a large number of features, the curse of dimensionality is imminent. To combat this issue, I chose to tackle the categorical features from multiple perspectives. Since the multiple-choice questions span across columns (number of columns equal to the number of choices in each question), simplification is done by first dropping the "other" columns within these questions because of the lack of information they provide. The B-part questions are then excluded because they ask about future plans and do not contain readily usable information for classification. Then each choice column is converted into 0 (NaN) or 1 (has the answer string), and separate columns are combined for each multiple-choice question (i.e. 12 parts of Q7 becomes a column of "number of programming languages used regularly").

For categorical columns containing NaN values such as Q8 and Q11, since these questions are mostly about the common use or recommendation of tools and methods, it made sense to fill these NaN values with the mode of the column (i.e. recommend learning "Python" first). Among the remaining categorical columns, ordinal numeric label mapping is applied to those with an ordinal nature such as age and education level. For nominal features, one-hot encoding is applied to preserve information without implying an order to the values. After data cleaning, there are 157 features left, which is easier to work with than the original 356.

EDA and Feature Selection

Fig 1. Top Ten Most Correlated Features

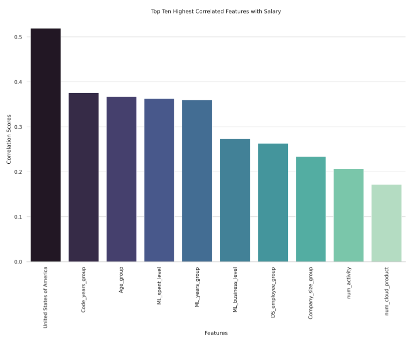
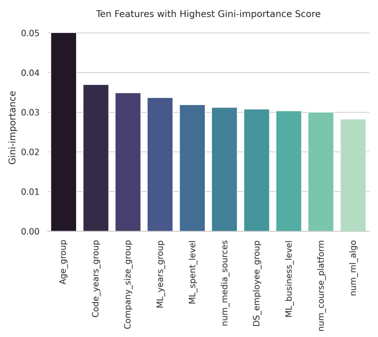


Fig 2. Features with Highest Gini-Importance



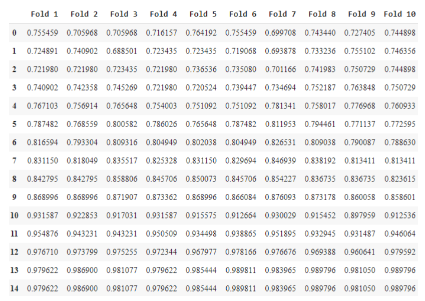
To capture features that matter the most, I first looked at their correlation with the target label (Q24_Encoded). As we can see from the correlation barplot (fig 1), the following features are among the most correlated with salary levels: whether the participant works in the USA; years of coding experience; age; the amount of money spent on machine learning services; and years of using machine learning methods. When a random forest classifier (which assigns Gini index to each feature in the process) is used as the first step of feature selection, it produces similar results as shown in fig 2 (though the US is not the most important feature here because one-hot encoding can be problematic in tree-based models). After selecting features with a positive Gini-importance score (155), Lasso CV is then applied to eliminate features with little relevance. The lasso-regularized method computes coefficients and applies a penalty to irrelevant features to bring their weights to 0. After selecting the remaining 103 features with nonzero coefficients, PCA is implemented next. Instead of looking at features directly, PCA computes the covariance matrix of the dataframe, then sorts the eigenvectors based on their eigenvalues. Results from PCA indicate that the first 28 principal components can explain about 95% of the variance within the data.

Model Implementation and Tuning

Before each model implementation, standardization is applied to both training and validation sets. This is because of differences in scales across independent features. Features undergo different types of encoding (ordinal versus one-hot encoding) and PCA transformation. Standardization can ensure that the features are within the same scale, thus better fit the model and predict targets.

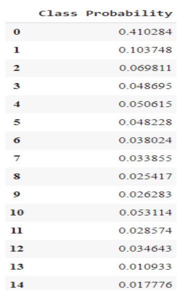
The first model implemented is the ordinal logistic regression. After standardization, 10-fold cross-validation, and incrementally classifying the targets into binary classes, the following accuracy, and probability tables are produced. Although the average scores across folds and across target groups are both around 70 to 90 percent, since they are calculated for cumulative class groups (i.e. class 0, class 0+1, class 0+1+2, etc.), these accuracies only show how dominant class 0 is compared to the other targets. The mean probability for each class is calculated by computing the mean probability for one target group (such as class 0+1+2), then subtracting the mean probability of the previous group from it. The detailed probabilities for each validation sample are also displayed in the notebook.

Table 1. Accuracies for Multi-class Labels across folds



	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10
0	0.755459	0.705968	0.705968	0.716157	0.764192	0.755459	0.699708	0.743440	0.727405	0.744898
1	0.724891	0.740902	0.688501	0.723435	0.723435	0.719068	0.693878	0.733236	0.755102	0.746356
2	0.721980	0.721980	0.723435	0.721980	0.736536	0.735080	0.701166	0.741983	0.750729	0.744898
3	0.740902	0.742358	0.745269	0.721980	0.720524	0.739447	0.734694	0.752187	0.763848	0.750729
4	0.767103	0.756914	0.765648	0.754003	0.751092	0.751092	0.781341	0.758017	0.776968	0.760933
5	0.787482	0.768559	0.800582	0.786026	0.765648	0.787482	0.811953	0.794461	0.771137	0.772595
6	0.816594	0.793304	0.809316	0.804949	0.802038	0.804949	0.826531	0.809038	0.790087	0.788630
7	0.831150	0.818049	0.835517	0.825328	0.831150	0.829694	0.846939	0.838192	0.813411	0.813411
8	0.842795	0.842795	0.858806	0.845706	0.850073	0.845706	0.854227	0.836735	0.836735	0.823615
9	0.868996	0.868996	0.871907	0.873362	0.868996	0.866084	0.876093	0.873178	0.860058	0.858601
10	0.931587	0.922853	0.917031	0.931587	0.915575	0.912664	0.930029	0.915452	0.897959	0.912536
11	0.954876	0.943231	0.943231	0.950509	0.934498	0.938865	0.951895	0.932945	0.931487	0.946064
12	0.976710	0.973799	0.975255	0.972344	0.967977	0.978166	0.976676	0.969388	0.960641	0.979592
13	0.979622	0.986900	0.981077	0.979622	0.985444	0.989811	0.983965	0.989796	0.981050	0.989796
14	0.979622	0.986900	0.981077	0.979622	0.985444	0.989811	0.983965	0.989796	0.981050	0.989796

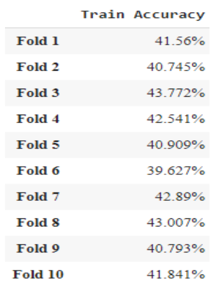
Table 2. Average Predicted Probabilities for each label



	Class Probability
0	0.410284
1	0.103748
2	0.069811
3	0.048695
4	0.050615
5	0.048228
6	0.038024
7	0.033855
8	0.025417
9	0.026283
10	0.053114
11	0.028574
12	0.034643
13	0.010933
14	0.017776

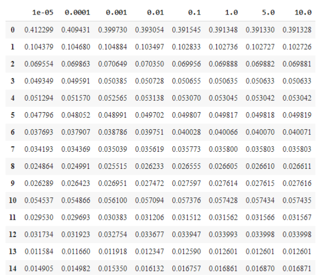
For comparison, models using multinomial logistic regression are also implemented. Since the results are averages across the entire model with multiple labels, the accuracy scores seem much lower (around 42% with a standard deviation of 1.2%). Treating each hyperparameter c value as a new model, the following probability table is produced. As we can see, the probabilities for each target label shown in this table are very similar to those produced by the ordinal logistic regression algorithm. Because of this, the next parts of the project will be carried out by using multinomial logistic regression due to its accessibility to tuning and prediction.

Table 3. Accuracies for each fold



	Train Accuracy
Fold 1	41.56%
Fold 2	40.745%
Fold 3	43.772%
Fold 4	42.541%
Fold 5	40.909%
Fold 6	39.627%
Fold 7	42.89%
Fold 8	43.007%
Fold 9	40.793%
Fold 10	41.841%

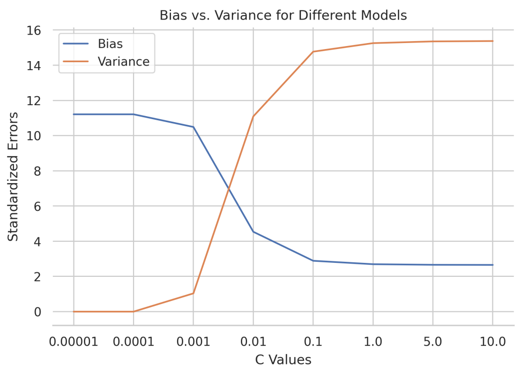
Table 4. Predicted Probabilities for each label with different C values



	1e-05	0.0001	0.001	0.01	0.1	1.0	5.0	10.0
0	0.412299	0.409431	0.399730	0.393054	0.391545	0.391348	0.391330	0.391328
1	0.104379	0.104680	0.104884	0.103497	0.102833	0.102736	0.102727	0.102726
2	0.069554	0.069863	0.070649	0.070350	0.069956	0.069888	0.069882	0.069881
3	0.049349	0.049591	0.050385	0.050728	0.050655	0.050635	0.050633	0.050633
4	0.051294	0.051570	0.052565	0.053138	0.053070	0.053045	0.053042	0.053042
5	0.047796	0.048052	0.048991	0.049702	0.049807	0.049817	0.049818	0.049819
6	0.037693	0.037907	0.038786	0.039751	0.040028	0.040066	0.040070	0.040071
7	0.034193	0.034369	0.035039	0.035619	0.035773	0.035800	0.035803	0.035803
8	0.024864	0.024991	0.025515	0.026233	0.026555	0.026605	0.026610	0.026611
9	0.026289	0.026423	0.026951	0.027472	0.027597	0.027614	0.027615	0.027616
10	0.054537	0.054866	0.056100	0.057094	0.057376	0.057428	0.057434	0.057435
11	0.029530	0.029693	0.030383	0.031206	0.031512	0.031562	0.031566	0.031567
12	0.031734	0.031923	0.032754	0.033677	0.033947	0.033993	0.033998	0.033998
13	0.011584	0.011660	0.011918	0.012347	0.012590	0.012601	0.012601	0.012601
14	0.014905	0.014982	0.015350	0.016132	0.016757	0.016861	0.016870	0.016871

Bias is the squared error between the average prediction of the model and the true target, while variance is the variability of the predictions. High bias indicates underfitting (left side of plot, heavy regularization, model is oversimplified), while high variance indicates overfitting (right side of plot, little regularization, model fits in noise within the train data, not generalizable). Smaller C values lead to simpler models that are prone to underfitting, (high bias, low variance) while larger C values lead to more complicated models that are easy to overfit (low bias, high variance). By treating each c value of hyperparameter as a new model, the bias-variance trade-off plot is constructed below. We can see that the balance occurs between c values of 0.001 and 0.01. Thus, we will be using this range of c values for tuning.

Fig 3. Bias-Variance Trade-off



Hyperparameters for logistic regression include regularization/penalty (L1, L2, Elastic net), C (inverse of regularization strength; smaller values specify stronger regularization), and solver types. However, since all solvers would produce the same results for convex functions (the cross-entropy cost function), this will not be considered for tuning. Since the 'saga' solver is suitable for large datasets with multiclass targets while supporting L1, L2, and elastic net penalties, we will be using this solver for tuning. F1 score is used in the grid search because of the presence of class imbalance within the multiclass dataset. Simply looking at accuracy might not be sufficient in this case because of different class frequencies. In order to obtain the best model, evaluating the F1 score is a good way to balance precision and recall. Here 'f1_weighted' is used because it calculates metrics for each label and finds their mean weighted by support (the number of true instances for each label). This should account for label imbalance in the data. After grid search, the best parameters are C=0.001, penalty='none' (could be caused by the small C value).

Testing and Discussion

Implementing the best model determined by grid search, the resulting training accuracy is 0.425 while the testing accuracy is 0.420. It is safe to say that the model only has minor overfitting issues. This is also shown by the learning curves in the notebook. Several reasons lead to the low accuracy. Information might be lost during feature engineering and encoding. Multiple choice questions are combined into a summation, while nominal features are one-hot encoded. Retaining more features might help save information, but the curse of dimensionality would be dire. The imbalance issue is apparent in the distribution plots as most of the participants are in the first salary level, which has more support in both training and testing reports. It is also possible that the multi-class logistic regression model did not capture the relationships within this data well. Ordinal logistic regression might do a better job.

Fig 4. Distribution of Target vs. Prediction in the train set

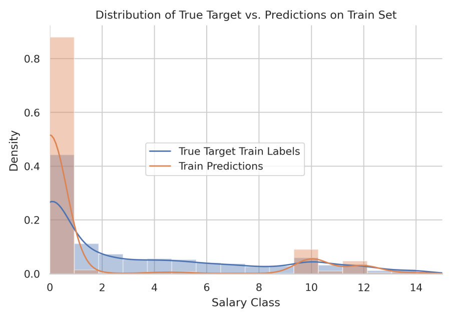
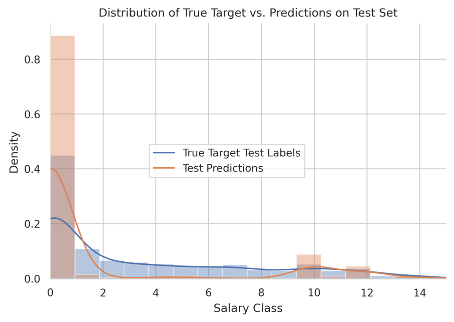


Fig 5. Distribution of Target vs. Prediction in the test set


Distribution of True Target vs. Predictions on Test Set
True Target Test Labels
Test Predictions
Density
Salary Class