MBTI PREDICTION MODEL

PERSONALITY TYPE FINDER

INDEX

Project Overview

Problem Statement

There is always a continuous and dynamic interaction between humans for various kinds of activities that are carried out in the world. Whether it is your average conversation with your friend or a meeting presentation with your boss, human interactivity is an integral part of daily activities that we carry out. Detecting the personality of a person can help us understand their behaviors, the choices they make and how they interact with everything around them. Knowing this can significantly increase the communication efficiency, optimize the use of resources, minimize conflicts and increase the efficiency of the tasks carried out.

One of the tools that is used currently for detecting the personality type of a person is Myers-Briggs Type Indicator (MBTI). This framework uses four dimensions to assess an individual which are as follows

- Extraversion (**E**) vs. Introversion (**I**): Focus of attention.
- Sensing (**S**) vs. Intuition (**N**): Information processing style.
- Thinking (**T**) vs. Feeling (**F**): Decision-making approach.
- Judging (**J**) vs. Perceiving (**P**): Approach to life and organization.

Based on various combinations of these four dimensions, there are 16 personality types. An individual can be categorized under one of these personality types. MBTI is typically calculated using well -articulated surveys, by interviews conducted by trained MBTI administrators or by observing an individual.

Machine learning for predicting MBTI

During our research we found out that using a machine learning algorithm for detecting the MBTI personality type can have several benefits over the conventional methodology. These benefits can be broadly classified as follows:

- **Efficiency and Scale:** The conventional methods like surveys or interviews are time consuming and costly, especially when intended to be used for a large number of people. Also, there can be human bias involved in using methods like interviews. Using a machine learning algorithm can eliminate these problems by making the process efficient without any biases and scalable on large datasets because of the high computational power available nowadays.

- **Additional Insights:** Conventional methods fail to capture the changes in the four dimensions of the MBTI framework with time. Since human personality is heavily influenced by experiences of individuals and their surroundings, they change over time. Using machine learning algorithms can help us understand and capture this trend. Additionally, it can also give us insights into more qualitative dimensions like age, gender, hobbies, etc that are helpful in determining someone's personality. This would give us more insights which a traditional human being analyzing the data might not be able to capture.
- **Predicting for Unassessed individuals:** The machine learning model can assess individuals without having to assess them, for example if your model is trained enough and if you know someones' age and gender, you can assess them after a certain period of time without having to do an reassessment to evaluate them.
- **Real-Time adoption:** Since we already know that human behavior doesn't remain constant and is influenced by various factors over time, we can always retrain our machine learning model with more data containing more dimensions that can adopt to the ever changing need.

This model can have a vast application, spanning across domains and industries since human personality is deeply rooted in various capacities in the work that is carried out. Some of the possible applications where this model can be helpful are as follows:

- Personalized recommendations for services: Career Counseling and Development, Personalized Learning recommendations, Product recommendations for individuals.
- Human Resources & Talent Management: For recruiting new employees, Team building activities.
- Mental Health & Well-being: Crafting therapies based on personality type, Stress management.
- Consumer Behavior Analysis: For effective market segmentation, Enhancing user experience of the product.
- Social and Communication Tools: Improving the social media interactions, Relationship compatibility

Goals

The primary goal of the project is to use the machine learning model to predict and classify an individual in one of the 16 personality types accurately. This goal can be further broken down into subgoals as follows

- Implement several machine learning models to the MBTI dataset

- Understand and analyze the relationship between the input (age, gender, education, introversion score, sensing score, thinking score, judging score and interest) and the output (personality type).
- Find out which input parameters are most relevant.
- Evaluate model performance and select the model with best performance. Measure how well each model works using metrics.

Success Metrics

We have selected Accuracy, Precision and Recall as the primary success metrics for this project. These three metrics provide a balanced and detailed evaluation of the model's performance in predicting the MBTI personality. The confusion matrix will provide us with the means to measure and calculate these three metrics.

- Accuracy: It will provide us the overall measure of how well our model is performing in detecting the MBTI personality type. High accuracy is a good and desirable measure. However, there are chances the accuracy alone can be misleading since there is a possibility of class imbalance in our dataset which will make it more accurate for our dataset but not accurate enough to predict the right personality.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision: This metric helps us understand how correctly is the model capable of predicting a particular type of personality. Having higher precision reduces the likelihood of false positives which means that a lower number of people will have their personality type classified wrongly.

$$Precision = \frac{TP}{TP + FP}$$

  We use the above formula to calculate the precision for each class, and the overall precision of the model is the mean of the class precisions.

- Recall: This metric gives the correct number of instances that our model was capable of identifying for all the personality types. It is important to have a high recall value so that we can be sure that all the personality types are adequately captured by our model.

$$Recall = \frac{TP}{TP + FN}$$

  We use the above formula to calculate the recall for each class, and the overall recall of the model is the mean of the class recalls.

Dataset and Methods

Dataset

Source - https://www.kaggle.com/datasets/stealthtechnologies/predict-people-personality-types

The dataset chosen has the purpose of exploring and predicting the Myers-Briggs type indicator (MBTI) personality types based on a combination of different elements, such as people's preferences, demographic factors and personality scores (rating on a scale of 1 to 10). Using these factors the personality is categorized in one of the 16 MBTI personalities.

The dataset has 128061 observations with 8 input parameters (x) and one output variable (y) which is the personality type.

The detailed description about the input parameters (x) is given below,

- **Age:** A continuous variable representing the age of the individual.
- **Gender:** A categorical variable indicating the gender of the individual. Possible values are 'Male' and 'Female'.
- **Education:** A binary variable, A value of 1 indicates the individual has at least a graduate-level education (or higher), and 0 indicates an undergraduate, high school level or Uneducated.
- **Interest:** A categorical variable representing the individual's primary area of interest.
- **Introversion Score:** A continuous variable ranging from 0 to 10, representing the individual's tendency toward introversion versus extroversion. Higher scores indicate a greater tendency toward extraversion.
- **Sensing Score:** A continuous variable ranging from 0 to 10, representing the individual's preference for sensing versus intuition. Higher scores indicate a preference for sensing.
- **Thinking Score:** A continuous variable ranging from 0 to 10, indicating the individual's preference for thinking versus feeling. Higher scores indicate a preference for thinking.

- **Judging Score:** A continuous variable ranging from 0 to 10, representing the individual's preference for judging versus perceiving. Higher scores indicate a preference for judging.

And the 1 response variable,

- **Personality**: A categorical variable representing 1 of the 16 types of MBTI.

There are no missing values in the dataset.

Methods

We checked the correlation between the parameters, and the covariance matrix between the input parameters is as follows,

	Age	Gender	Education	Introversion.Score	Sensing.Score	Thinking.Score	Judging.Score	Interest
Age	24.089	0.004	0.009	-0.013	-0.019	0.017	0.003	-0.010
Gender	0.004	0.248	0.000	-0.005	0.000	-0.006	-0.000	-0.001
Education	0.009	0.000	0.242	-0.003	-0.002	0.004	0.001	0.002
Introversion.Score	-0.013	-0.005	-0.003	8.344	-0.002	0.023	0.003	0.005
Sensing.Score	-0.019	0.000	-0.002	-0.002	1.400	-0.003	-0.003	-0.005
Thinking.Score	0.017	-0.006	0.004	0.023	-0.003	8.329	0.008	0.019
Judging.Score	0.003	-0.000	0.001	0.003	-0.003	0.008	1.936	0.002
Interest	-0.010	-0.001	0.002	0.005	-0.005	0.019	0.002	2.522

As it is evident from the matrix that the non-diagonal values are not zero that means there is a correlation between the parameters. Also, the diagonal values are different signifying that the distribution might not be a normal distribution.

In the given scenario, we decided to work on our data applying 4 different methods:

1. Logistic regression

Using logistic regression, the idea is to model our categorical variable (personality). It is generally applied in classification tasks, as we want to define to which category the combination of our predictors for each observation belongs to. Moreover, considering that we can obtain 16 different types of personalities, we employed the multinomial logistic regression, assuming the independence among the observations and non-perfect multicollinearity among predictors. Finally, the main advantages of this method for our purpose are related to an easy implementation and interpretation, and to the ability of handling both categorical and continuous variables.

2. Lasso Regression

It is a typology of linear regression used to improve the normal linear regression model through the incorporation of a penalty term. The main goal of this algorithm is to select the important features in the dataset and reduce the complexity. Adding the penalty term, we increase the bias but we reduce the variance, improving generalization to the

new data. The main advantages are the automatic selection of important features, shrinking coefficients of irrelevant terms to zero, improved interpretability of the model and the prevention from overfitting.

3. Ridge regression

Ridge regression is another linear regression algorithm, focusing on multicollinearity and overfitting; the idea is to add a regularization term to the OLS loss function and, differently from Lasso, all predictors are kept into the model. This method is particularly useful when predictors are highly correlated because it distributes the values of coefficients among correlated predictors, reducing the variance. As in the Lasso case, it introduces bias, but it reduces variance.

4. Decision tree

The decision tree method is a supervised learning algorithm that we used to split the dataset into subsets based on feature values, creating a tree-like structure that is then used to make decisions. Through this method, we continue to split the dataset in smaller and smaller subsets to reduce the impurity of the results and to maximize the homogeneity in the same subsets; this process is carried on until a stopping condition is met. The main advantages here are the intuitive and easy-to-interpret structure, no need for assumptions on the data distribution and that it provides a natural way to determine the feature significance.

5. Random forest

Random forest is a supervised learning ensemble method employed for both classification and regression. It constructs forests of multiple decision trees during the training process and combines their predictions to get better accuracy and reduce overfitting. Each tree is grown independently from the others using the bootstrapped dataset so that they can avoid pruning, since single trees will overfit the data. In our case (classification), each tree gives a response in terms of personality, and the majority of votes for a personality will give the final prediction for the “forest”. The main advantages of this algorithm are the increased accuracy (thanks to the combination of multiple trees), reduced overfitting risk, and the generation of insights on which features are most important for predictions.

6. Support vector machine

SVM is another supervised learning algorithm that suits particularly well for high-dimensional data and complex relationships between features and labels. The idea behind this method is to build hyperplane or non-linear boundaries to separate the different classes in the space. The main advantages are the effectiveness in high-dimensional spaces, the versatility (being able to work with linear and non-linear data), and the ability to reduce or avoid overfitting. On the other hand, we can say it has some limitations in terms of computation efficiency and it requires a careful selection of kernels and hyperparameters.

Our Approach

We started by inspecting the dataset that we have for our MBTI model. The dataset has both categorical and continuous variables. There are no missing values for any of the parameters. Hence, the only operations that were required for the dataset were scaling the parameters whenever required to make them compatible to use.

In the next step, we tried to determine if the parameters were correlated to each other and to what extent. We used a correlation matrix for this and found out that the parameters were weakly correlated with each other. This is because each parameter is a unique trait in an individual's personality, which is distinct from one another. We have used K-fold validation for dividing the dataset into training and testing data for each model. Since we have an imbalanced dataset, where some classes have few instances, we use 5 folds in cross validation to ensure that there are enough observations of each class in every fold.

We started with a simple method, Logistic Regression, which has a linear decision boundary and is easily interpretable. We used this method to estimate two models, one with normal scaling of parameters and other with PCA. Comparing these methods helped us understand the trade-off required between scaling of features and reducing the dimensionality of the features. We observed that scaling the features yields better results than reducing the dimensions of the features because naturally, in determining the personality trait of a person, all the features are important, and the dataset is not high-dimensional.

In the next step we applied more linear methods like Lasso Regression and Ridge Regression that were better with handling multicollinearity effectively. Then we moved to non-linear methods like Decision trees and Random forests because of their ability to handle the complex relationships among the parameters and make a model which is a better fit. Then we tried Support Vector Machines (SVM) methods (linear, polynomial and radial kernels) to test different linear and non-linear decision boundaries.

The purpose of trying out the above-mentioned methods was to understand the complex relationships among the distinct parameters and measure them using our success metrics. This would allow us to choose a single good method that gives us a good model to predict the MBTI personality type.

Results and Challenges

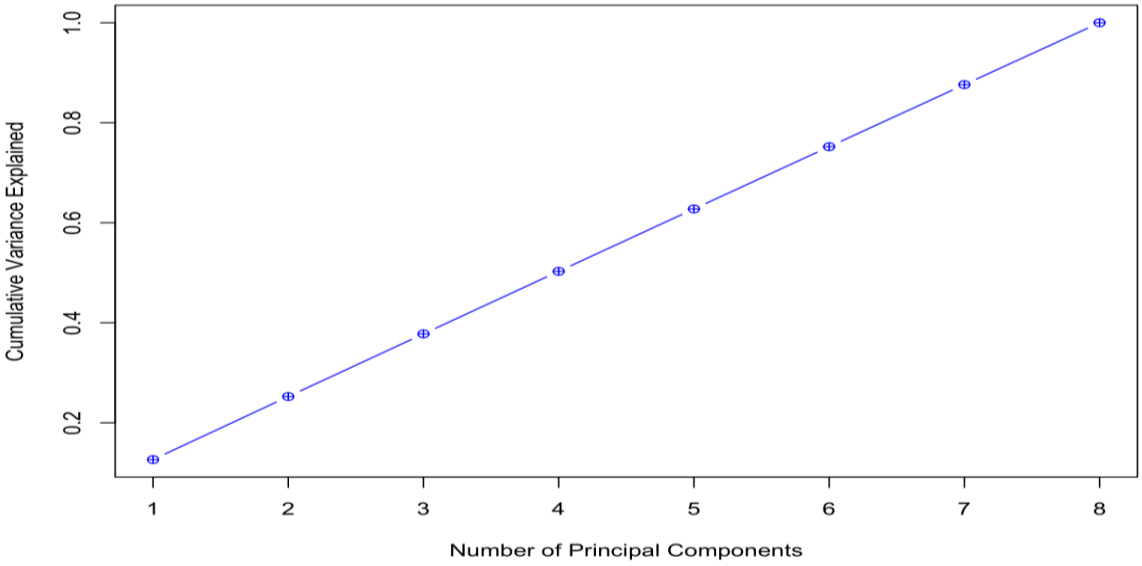
Results

We used k-folds cross validation on every model that we applied. This reduces the variance and enables us to get a more robust estimation on the model's performance.

1. Logistic Regression

We applied PCA to visualize the relationships between the features and to identify any potential features that could be combined.

Cumulative Explained Variance Plot for PCA



Since no clear elbow is observed, indicating that all components are similarly significant in representing the variance, we will use all 8 principal components. Given that the number of features in this dataset is relatively small, it is reasonable to use all principal components.

(with Principal Component Analysis)

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	135	30	24	41	20	3	0	0	0	0	0	0	0	0	0	0
1	670	6578	0	109	0	0	0	0	0	0	0	0	0	0	0	0
2	13	0	202	66	1	0	0	0	0	0	0	0	0	0	0	0
3	0	84	290	4620	0	0	0	1	0	0	0	0	0	0	0	0
4	9	7	3	8	53	30	0	0	11	0	0	0	0	0	0	0
5	0	131	0	5	37	722	0	10	5	26	0	0	0	0	0	0
6	0	0	1	9	4	16	56	89	5	16	19	0	0	0	0	0
7	0	0	3	118	2	72	15	561	1	28	0	0	0	0	0	0
8	0	0	2	0	6	1	0	0	420	118	6	0	0	0	0	0
9	0	0	0	0	0	66	0	3	124	4589	0	0	0	0	0	0
10	0	0	0	0	0	1	7	13	12	157	211	7	0	0	0	0
11	0	0	0	0	0	0	0	0	0	10	136	3131	0	3	15	182
12	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	4	0	9	56	550	0	0
14	0	0	0	0	0	0	0	1	0	13	0	1	0	0	0	0
15	0	0	0	0	0	0	0	0	0	5	0	325	19	96	33	307

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	96	9	19	33	31	62	0	6	0	0	0	0	0	0	0	0
1	665	6726	0	2	0	31	0	0	0	0	0	0	0	0	0	0
2	8	3	304	88	2	0	28	19	0	0	0	0	0	0	0	0
3	6	117	261	4744	0	17	1	39	0	0	0	0	0	0	0	0
4	4	3	10	14	56	34	40	22	22	3	2	0	0	0	0	0
5	2	18	0	11	26	812	0	15	2	64	0	0	0	1	0	0
6	0	0	0	0	0	1	0	1	0	3	0	0	0	0	0	0
7	0	0	0	14	0	4	23	564	0	24	1	31	0	0	0	0
8	0	0	0	0	0	0	0	0	380	292	16	2	0	0	0	0
9	0	0	0	0	0	2	0	0	126	4480	0	0	0	0	0	0
10	0	0	0	0	0	0	0	1	1	8	235	1	0	0	0	0
11	0	0	0	0	0	0	0	0	0	6	92	3212	0	1	10	116
12	0	0	0	0	0	0	0	0	21	60	18	11	22	2	1	0
13	0	0	0	0	0	0	0	0	0	5	0	4	33	615	0	0
14	0	0	0	0	0	0	3	10	0	1	20	5	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	147	4	97	52	355

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	525	193	31	11	44	5	0	0	0	0	0	0	0	0	0	0
1	264	6366	0	1	0	0	0	0	0	0	0	0	0	0	0	0
2	10	33	223	136	2	9	7	5	0	0	0	0	0	0	0	0
3	0	180	292	4748	0	5	3	42	0	0	0	0	0	0	0	0
4	0	0	0	0	2	3	0	0	0	0	0	0	0	0	0	0
5	2	85	0	1	22	859	0	1	5	92	0	0	0	0	0	0
6	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0
7	0	0	0	56	0	61	17	568	0	1	0	1	0	0	0	0
8	0	0	0	0	0	12	2	3	508	514	9	0	0	0	0	0
9	0	0	0	0	0	0	0	0	63	4165	0	5	0	0	0	0
10	0	0	0	0	0	0	0	0	14	31	271	8	0	0	0	0
11	0	0	0	0	0	0	0	0	0	33	90	3081	0	0	23	170
12	0	0	1	0	31	31	33	29	18	16	1	0	1	0	0	0
13	0	0	0	0	0	0	0	0	0	20	0	4	77	622	3	41
14	0	0	0	0	0	0	0	5	0	3	9	9	0	0	1	0
15	0	0	0	0	0	0	0	0	0	63	1	326	7	72	23	251

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	106	159	4	47	40	46	0	1	6	11	0	0	0	0	0	0
1	571	6552	0	8	0	0	0	0	0	0	0	0	0	0	0	0
2	57	34	260	134	1	0	0	0	0	0	0	0	0	0	0	0
3	0	82	276	4652	0	6	0	56	0	0	0	0	0	0	0	0
4	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
5	7	89	0	10	58	882	0	22	1	67	0	0	0	0	0	0
6	0	0	17	7	15	7	58	17	0	2	6	1	0	0	0	0
7	0	0	5	80	0	46	19	572	0	1	1	12	0	0	0	1
8	0	0	0	0	0	0	0	2	448	466	18	17	1	0	0	0
9	0	0	0	0	0	6	0	0	143	4139	0	23	0	0	0	0
10	0	0	0	0	0	0	0	1	17	52	314	63	0	0	0	0
11	0	0	0	0	0	0	0	0	0	86	52	3044	0	0	16	168
12	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	12	57	621	1	14
14	0	0	0	0	0	0	0	3	2	10	13	30	11	1	8	0
15	0	0	0	0	0	0	0	0	0	35	0	199	5	67	29	301

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	46	17	31	21	19	27	0	2	3	18	0	0	0	0	0	0
1	668	6825	0	17	0	3	0	0	0	0	0	0	0	0	0	0
2	9	10	195	90	0	0	8	7	14	35	9	0	0	0	0	0
3	0	42	319	4750	0	19	2	77	0	11	0	0	0	0	0	0
4	7	2	5	2	59	77	11	2	6	10	0	0	5	0	0	0
5	2	29	1	18	20	822	0	6	2	85	0	0	0	2	0	0
6	0	0	4	0	1	0	0	1	1	0	0	0	0	0	0	0
7	0	0	0	47	0	26	59	554	0	16	1	34	0	1	0	1
8	0	0	0	0	0	1	0	0	367	230	23	1	3	0	0	0
9	0	0	0	0	0	5	0	0	146	4433	0	3	8	4	0	0
10	0	0	0	0	0	0	0	1	24	56	281	2	0	0	0	0
11	0	0	0	0	0	0	0	0	0	41	59	3232	0	3	19	204
12	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0
13	0	0	0	0	0	0	0	0	0	25	0	11	44	535	0	0
14	0	0	0	0	0	0	0	1	0	12	5	17	6	3	1	0
15	0	0	0	0	0	0	0	0	0	2	0	111	11	160	27	279

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Overall
Accuracy	0.86	0.88	0.87	0.86	0.87	0.87
Precision	0.60	0.62	0.59	0.61	0.59	0.60
Recall	0.61	0.62	0.59	0.62	0.58	0.60

Precision breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.46	0.92	0.59	0.91	0.42	0.81	0.14	0.78	0.57	0.97	0.76	0.91	0.10	0.88	0.03	0.44

Recall breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.23	0.96	0.77	0.95	0.31	0.85	0.29	0.84	0.73	0.88	0.68	0.92	0.4	0.85	0.04	0.62

(with just scaling)

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	549	118	57	7	25	0	0	0	0	0	0	0	0	0	0	0
1	276	6660	0	7	1	0	0	0	0	0	0	0	0	0	0	0
2	1	2	374	46	2	0	28	0	0	0	0	0	0	0	0	0
3	0	28	92	4882	2	13	9	60	0	4	0	0	0	0	0	0
4	0	1	2	0	45	4	1	0	4	0	0	0	0	0	0	0
5	1	21	0	19	36	862	3	36	3	153	0	0	0	0	0	0
6	0	0	0	4	4	1	12	5	0	0	0	0	0	0	0	0
7	0	0	0	7	0	5	21	571	0	5	3	61	2	2	1	28
8	0	0	0	4	8	8	0	1	430	253	55	11	33	2	0	0
9	0	0	0	0	0	18	0	0	138	4248	0	0	6	28	0	0
10	0	0	0	0	0	0	4	0	2	0	193	1	0	0	3	0
11	0	0	0	0	0	0	0	0	5	1	307	121	3384	1	48	23	186
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	7	0	0	30	546	1	44
14	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	16	3	23	20	231

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	522	99	76	10	23	0	0	0	0	0	0	0	0	0	0	0
1	258	6716	0	5	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	387	50	6	0	28	0	0	0	0	0	0	0	0	0
3	0	29	129	4799	0	12	18	58	1	0	0	0	0	0	0	0
4	0	1	0	1	31	12	0	0	3	0	0	0	0	0	0	0
5	1	31	0	20	48	905	3	40	3	147	0	0	0	0	0	0
6	0	0	2	4	2	3	8	6	0	0	0	1	0	0	0	0
7	0	0	0	9	0	1	28	567	0	10	3	55	1	3	6	18
8	0	0	0	8	5	11	0	1	398	241	57	13	24	5	1	0
9	0	0	0	0	0	19	0	0	145	4246	0	0	2	28	0	0
10	0	0	0	0	0	0	10	1	2	0	186	1	0	0	7	0
11	0	0	0	0	0	0	0	4	0	290	138	3326	3	55	15	194
12	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
13	0	0	0	0	0	0	0	0	0	9	0	0	25	595	0	56
14	0	0	0	0	0	0	0	0	0	3	0	1	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	16	3	30	34	203

Actual															
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	558	99	60	6	17	0	0	0	0	0	0	0	0	0	0	0
1	243	6696	0	6	3	0	0	0	0	0	0	0	0	0	0	0
2	0	0	385	53	3	0	12	0	0	0	0	0	0	0	0	0
3	0	29	102	4850	2	14	6	49	0	1	0	0	0	0	0	0
4	0	1	0	0	27	10	2	0	3	0	0	0	0	0	0	0
5	0	32	0	16	41	923	3	28	7	122	0	0	0	0	0	0
6	0	0	0	7	1	1	8	1	0	1	1	0	0	0	0	0
7	0	0	0	7	0	4	24	568	0	4	1	79	2	7	3	34
8	0	0	0	8	7	4	2	2	461	210	39	10	29	1	0	0
9	0	0	0	0	0	29	0	0	133	4285	0	1	8	21	0	0
10	0	0	0	0	0	0	5	0	4	0	200	0	0	0	4	0
11	0	0	0	0	0	0	0	6	0	308	140	3329	5	49	22	170
12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	7	0	0	41	588	2	60
14	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	14	0	28	19	198

Actual															
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	504	116	57	9	19	0	0	0	0	0	0	0	0	0	0	0
1	238	6737	0	4	3	0	0	0	0	0	0	0	0	0	0	0
2	0	0	391	41	4	0	27	0	0	0	0	0	0	0	0	0
3	0	31	114	4830	2	9	11	58	0	2	0	0	0	0	0	0
4	0	0	0	0	41	12	1	0	2	2	0	0	0	0	0	0
5	0	32	0	28	38	947	3	32	1	153	0	0	0	0	0	0
6	0	0	0	5	0	3	8	4	0	1	0	0	1	0	0	0
7	0	0	0	16	0	2	21	574	0	1	5	61	2	8	6	24
8	0	0	0	5	9	6	4	0	465	247	53	11	30	2	0	0
9	0	0	0	0	0	14	0	0	149	4135	0	0	3	32	0	0
10	0	0	0	0	0	0	2	0	0	0	204	0	0	0	3	0
11	0	0	0	0	0	0	0	6	0	321	142	3313	3	58	20	177
12	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	5	0	0	31	567	2	59
14	0	0	0	0	0	0	0	0	0	2	0	0	1	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	16	3	22	23	224

Actual																
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	510	117	69	11	24	0	0	0	0	0	0	0	0	0	0	0
1	221	6759	0	9	4	0	0	0	0	0	0	0	0	0	0	0
2	1	0	386	39	3	0	27	0	0	0	0	0	0	0	0	0
3	0	27	100	4829	0	19	8	64	2	1	0	0	0	0	0	0
4	0	1	0	1	27	7	0	0	2	1	0	0	0	0	0	0
5	0	21	0	31	37	922	6	32	2	140	0	0	0	0	0	0
6	0	0	0	1	2	1	9	3	0	0	0	0	0	0	0	0
7	0	0	0	19	0	2	28	546	0	5	1	55	2	9	2	30
8	0	0	0	5	2	9	1	3	421	239	39	15	30	1	0	0
9	0	0	0	0	0	20	0	0	132	4280	0	0	8	33	0	0
10	0	0	0	0	0	0	1	2	2	0	213	1	1	0	6	0
11	0	0	0	0	0	0	0	1	3	302	125	3334	4	69	14	185
12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	2	0	0	32	556	0	41
14	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	1	6	1	40	25	228

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Overall
Accuracy	0.89	0.89	0.90	0.89	0.89	0.89
Precision	0.75	0.73	0.69	0.69	0.70	0.71
Recall	0.63	0.61	0.62	0.62	0.62	0.62

Precision breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.75	0.69	0.73	0.65	0.55	0.60	0.64	0.73	0.81	0.86	0.77	0.75	0.47	0.50	0.37	0.45

Recall breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.82	0.66	0.78	0.71	0.60	0.56	0.67	0.78	0.9	0.93	0.85	0.7	0.53	0.55	0.6	0.51

When performing logistic regression, applying it on the 8 PCs obtained after PCA performs slightly better than applying on the raw features. However, both perform very well in predicting the 16 classes.

2. Lasso Regression

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	617	50	12	4	32	4	0	0	44	4	1	0	3	0	0	0
2	105	6090	0	99	2	303	0	5	9	376	0	8	2	19	0	0
3	10	0	439	37	0	0	16	4	1	1	18	2	0	0	4	0
4	4	123	67	4379	0	4	3	254	0	7	11	278	0	1	0	18
5	14	0	1	0	55	1	0	0	0	0	0	0	7	0	0	0
6	6	177	0	6	14	574	0	9	1	10	0	0	1	33	0	0
7	0	0	8	2	0	0	42	7	0	0	0	1	0	0	0	0
8	0	5	1	117	0	8	2	349	0	0	0	9	0	0	0	22
9	40	5	2	0	3	0	0	0	445	33	11	1	24	3	1	0
10	2	392	0	7	0	23	0	0	71	4295	0	86	4	227	0	4
11	1	0	32	4	0	0	2	1	6	0	270	28	0	0	14	1
12	0	5	3	270	0	1	0	11	4	87	39	2927	0	7	3	180
13	1	0	0	0	3	0	0	0	15	0	0	0	33	1	0	0
14	0	10	0	1	0	35	0	0	2	149	0	3	10	394	0	4
15	0	0	1	0	0	0	3	2	1	0	13	0	1	0	16	0
16	0	0	0	8	0	0	0	16	0	0	0	93	0	10	7	253

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	560	75	17	4	33	3	0	0	29	3	0	0	5	0	0	0
2	113	6120	0	137	4	336	0	5	8	356	0	5	0	19	0	0
3	11	1	418	51	2	0	25	4	0	0	33	4	0	0	1	0
4	2	145	61	4356	0	11	3	259	1	5	3	291	0	0	0	19
5	27	0	0	0	44	1	0	0	0	0	0	0	6	0	1	0
6	1	156	0	4	6	504	0	9	1	8	0	0	0	37	0	0
7	1	0	13	1	1	0	40	3	0	0	2	0	0	0	0	0
8	0	2	1	130	2	15	6	348	0	0	0	5	0	0	2	27
9	29	4	1	0	5	2	0	0	483	48	12	1	22	4	1	0
10	12	375	0	3	0	14	0	0	68	4315	0	62	1	214	0	8
11	2	0	18	2	0	0	2	1	5	0	319	17	0	0	11	0
12	0	10	4	272	0	0	0	13	4	89	54	2928	0	4	4	157
13	1	1	0	0	12	0	0	0	8	2	0	0	36	0	0	0
14	0	8	0	0	0	32	0	1	0	134	0	3	6	373	0	6
15	1	0	1	0	0	0	5	0	0	0	7	1	1	0	29	4
16	0	0	0	7	0	4	0	26	0	0	0	86	0	12	3	260

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	599	48	14	0	30	2	0	0	38	6	0	0	6	0	0	0
2	105	6096	0	149	3	352	0	6	5	378	0	9	0	29	0	1
3	15	0	421	38	0	0	25	1	0	1	29	3	0	0	1	0
4	4	134	72	4351	0	8	3	257	0	6	4	251	0	2	1	14
5	26	7	1	0	55	10	1	0	1	0	0	0	4	1	0	0
6	1	194	0	2	2	604	0	11	0	14	0	0	1	32	0	0
7	0	0	11	0	0	0	41	2	0	0	0	0	0	0	6	0
8	1	4	3	123	0	8	9	373	0	0	1	10	0	0	0	16
9	39	5	0	0	2	0	0	0	421	45	8	4	29	1	2	0
10	5	346	0	6	0	19	0	0	65	4242	0	102	2	282	0	5
11	1	0	32	4	0	0	0	0	9	2	270	29	0	0	15	1
12	0	13	3	262	0	3	0	13	6	65	49	2893	0	4	1	167
13	2	1	0	0	3	0	0	0	13	1	0	0	23	2	0	0
14	1	11	0	0	1	35	0	1	1	115	0	5	8	375	0	11
15	0	0	0	0	0	0	2	1	1	0	9	1	0	0	19	4
16	0	0	0	8	0	0	0	28	0	2	1	83	1	9	4	266

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	605	59	3	0	41	4	1	0	43	4	0	0	4	0	0	0
2	105	6145	0	111	3	313	0	7	5	371	0	3	1	16	0	0
3	10	0	440	45	0	0	20	1	0	0	22	2	0	0	3	0
4	6	107	71	4333	0	8	2	245	0	5	6	249	0	1	2	16
5	25	0	1	0	50	4	1	0	2	0	0	0	3	0	0	0
6	2	179	0	1	6	532	0	7	1	7	0	1	2	27	0	0
7	0	0	16	1	2	0	38	4	0	0	2	0	0	0	6	0
8	1	3	2	141	1	12	4	368	0	1	0	10	0	1	0	29
9	29	6	1	0	2	1	0	0	429	32	6	3	20	1	0	0
10	1	354	0	7	0	27	0	0	63	4372	0	74	4	240	0	2
11	1	0	19	2	0	0	1	0	8	0	288	31	0	0	21	1
12	2	8	6	290	0	0	0	6	3	91	45	2974	0	5	1	152
13	2	0	0	0	4	2	1	0	13	0	1	0	33	4	2	0
14	0	7	0	0	0	33	0	1	3	115	0	1	4	393	0	7
15	0	0	0	0	0	0	3	0	0	0	4	0	1	0	19	2
16	0	0	0	10	0	2	0	22	0	1	0	83	0	3	8	267

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	550	64	14	2	42	8	2	0	38	5	1	1	3	1	0	0
2	102	6197	0	132	9	315	0	10	8	351	0	9	2	20	0	0
3	10	1	431	41	3	0	33	1	0	0	31	2	0	0	3	0
4	6	117	73	4374	0	6	4	231	0	8	4	277	0	2	0	15
5	15	0	0	0	65	5	0	0	0	0	0	0	1	0	0	0
6	2	188	0	3	8	580	0	11	0	14	0	1	0	24	0	0
7	0	0	10	1	1	0	45	2	0	0	0	0	0	0	1	0
8	0	2	2	115	0	7	6	355	0	0	0	6	0	3	0	21
9	45	5	0	0	3	0	0	0	448	51	8	0	14	4	0	0
10	4	336	0	9	0	27	0	2	64	4275	0	102	5	230	0	8
11	1	0	28	0	0	0	0	0	10	1	279	30	0	0	22	2
12	0	2	8	243	0	1	0	15	5	79	56	2939	0	3	1	162
13	0	0	0	0	0	0	0	0	7	2	0	0	33	2	0	0
14	0	11	0	0	0	29	0	0	3	125	0	3	4	374	0	4
15	0	0	1	2	0	0	1	1	0	0	3	0	1	0	21	5
16	0	0	0	11	0	0	0	25	0	2	0	102	0	7	6	249

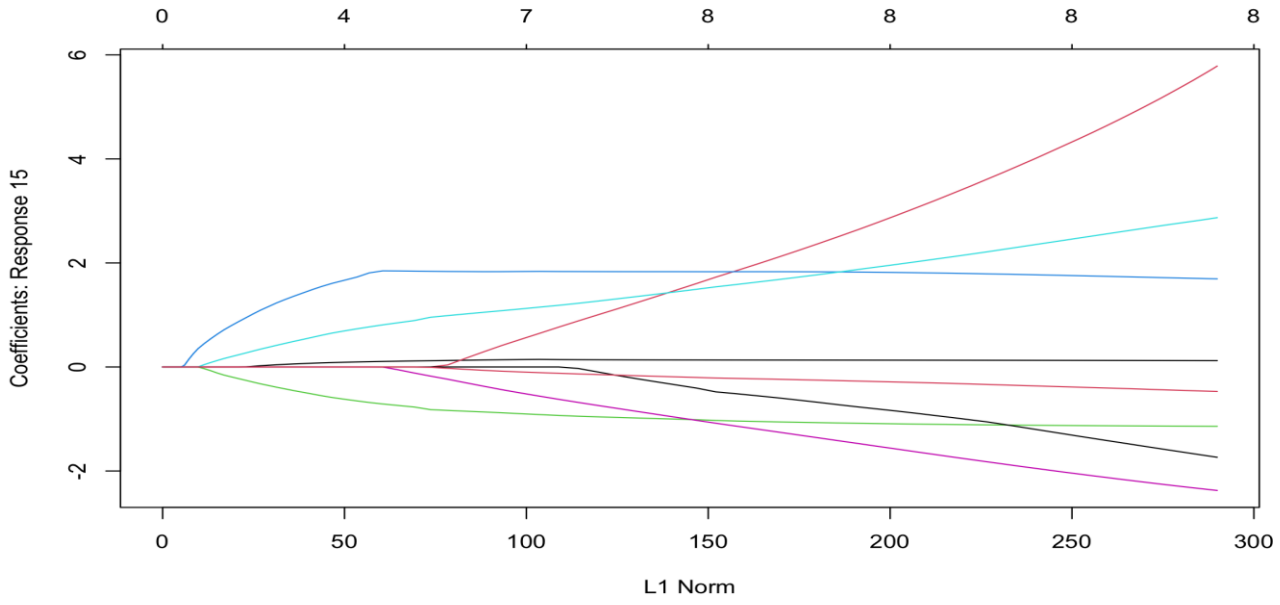
	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Overall
Accuracy	0.83	0.83	0.82	0.83	0.83	0.83
Precision	0.73	0.73	0.71	0.73	0.75	0.73
Recall	0.66	0.67	0.65	0.66	0.66	0.66

Precision breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.76	0.89	0.77	0.88	0.50	0.58	0.53	0.54	0.76	0.87	0.74	0.85	0.35	0.55	0.41	0.54

Recall breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.76	0.89	0.77	0.62	0.69	0.68	0.67	0.67	0.78	0.85	0.77	0.83	0.60	0.67	0	0.65



As we move along the x-axis, none of the features shrink to zero; instead, they all have larger coefficients. This indicates that all features have a strong relationship with the response variable.

```
Average number of selected features in Lasso: 8
```

And the number of features selected in all 5 folds is all 8 features. This indicates that all the features are relevant.

3. Ridge Regression

	Actual														
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	92	1	19	0	23	0	2	0	18	0	1	0	3	0	0	0
1	688	6308	10	183	68	719	0	12	32	454	0	6	2	31	0	0
2	0	0	27	0	0	0	5	0	1	0	5	0	0	0	1	0
3	6	99	487	4456	1	13	62	557	1	7	21	337	0	3	4	46
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	10	27	1	3	29	183	5	26	2	4	0	0	8	34	0	3
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	2	8	0	0	18	51	0	0	0	1	0	0	1	17
8	3	0	0	0	3	0	0	0	31	1	7	0	8	0	3	0
9	34	376	0	9	2	44	0	0	440	4408	8	110	45	554	2	11
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	1	12	28	267	0	1	3	26	13	67	325	2952	2	15	40	386
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	1	0	0	2	5	0	0	5	4	1	2	13	76	2	14
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2	4

	Actual															
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	81	0	12	0	10	0	4	0	20	0	3	0	6	0	1	0
1	660	6352	9	169	60	712	0	10	39	424	0	8	3	36	0	0
2	0	0	24	0	0	0	1	0	0	0	10	1	0	0	2	0
3	17	127	455	4522	1	21	48	544	1	7	33	344	0	4	3	36
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	5	32	0	6	18	187	2	24	0	3	0	0	3	38	0	2
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	1	7	0	0	9	60	0	0	1	3	0	0	3	20
8	6	0	0	0	0	0	0	0	32	0	11	0	9	0	1	0
9	27	335	0	8	4	38	0	0	445	4477	6	125	49	526	1	8
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	1	12	28	260	0	0	2	36	14	66	341	2898	0	11	27	414
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	1	1	0	0	1	7	0	2	5	14	1	2	10	78	1	11
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	2	0	0	1	13

	Actual															
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	91	0	6	0	16	1	2	0	24	0	3	0	3	0	0	0
1	589	6433	11	148	63	732	1	14	35	432	0	5	3	38	0	0
2	0	0	16	0	0	0	9	0	0	0	7	0	0	0	2	0
3	9	100	464	4558	1	12	48	542	0	16	34	320	0	0	2	29
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	9	26	0	4	35	182	1	37	0	6	0	0	3	32	1	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	3	0	0	13	47	0	0	0	5	0	0	4	13
8	2	0	1	0	0	0	0	0	27	0	4	0	7	0	2	0
9	29	354	1	7	2	32	0	0	476	4420	6	145	40	543	0	9
10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
11	3	11	40	240	0	2	1	24	12	64	339	2951	4	8	43	398
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	1	0	0	0	1	4	0	2	3	9	0	5	12	63	1	7
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	1	0	0	0	0	0	0	0	0	1	0	0	2	10

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	103	0	23	0	22	0	3	0	26	0	4	0	3	0	1	0
1	615	6344	10	185	47	689	1	15	41	385	0	5	4	31	0	0
2	0	0	22	0	0	0	4	0	0	0	9	0	0	0	4	0
3	15	113	476	4472	0	11	52	548	0	13	22	310	0	1	3	34
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	3	39	0	5	27	187	1	25	3	4	0	0	7	38	0	1
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	4	7	0	0	11	46	0	1	0	4	0	0	1	14
8	1	0	0	0	2	0	0	0	22	0	3	0	1	0	3	0
9	38	377	0	14	4	65	0	1	518	4434	7	130	38	558	0	16
10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
11	3	5	22	259	0	1	3	30	15	69	339	2985	0	11	38	373
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	2	0	0	1	7	0	1	1	13	0	3	11	63	2	18
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	1	0	0	0	0	0	0	0	0	0	0	3	7

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	73	2	15	0	24	0	1	0	20	0	2	0	5	0	0	0
1	613	6361	7	157	55	707	1	8	38	400	0	7	4	38	0	0
2	0	0	19	0	0	0	4	0	0	0	2	0	0	0	1	0
3	11	117	508	4504	3	13	48	522	0	7	30	313	0	1	5	28
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	7	33	1	4	23	201	2	35	1	4	0	1	6	33	0	3
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	5	6	0	0	21	51	0	0	1	0	0	0	2	19
8	0	0	0	0	1	0	1	0	22	0	9	0	5	0	1	0
9	34	395	1	8	4	45	0	0	523	4428	11	115	42	507	0	8
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	1	7	27	238	0	5	0	37	9	58	312	3031	2	12	40	402
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	1	2	0	0	1	6	1	1	4	12	0	3	10	73	2	9
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	2	0	0	4	6

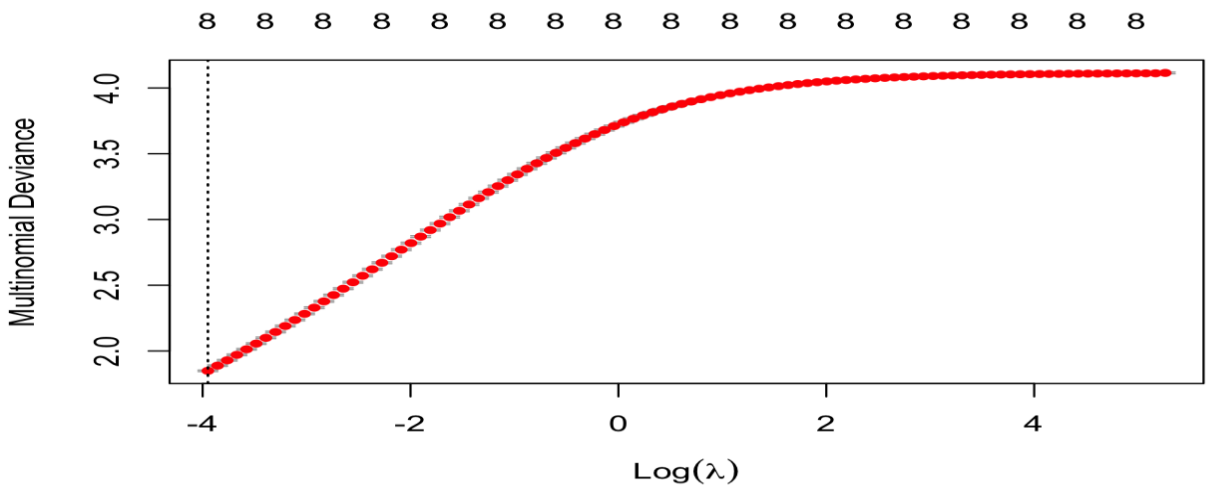
	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Overall
Accuracy	0.73	0.73	0.73	0.73	0.73	0.73
Precision	0.63	0.66	0.67	0.66	0.62	0.65
Recall	0.26	0.26	0.26	0.26	0.26	0.26

Precision breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.57	0.76	0.62	0.74	0	0.56	0	0.53	0.60	0.9	0	0.86	0	0.10	0	0.00

Recall breakdown for each class:

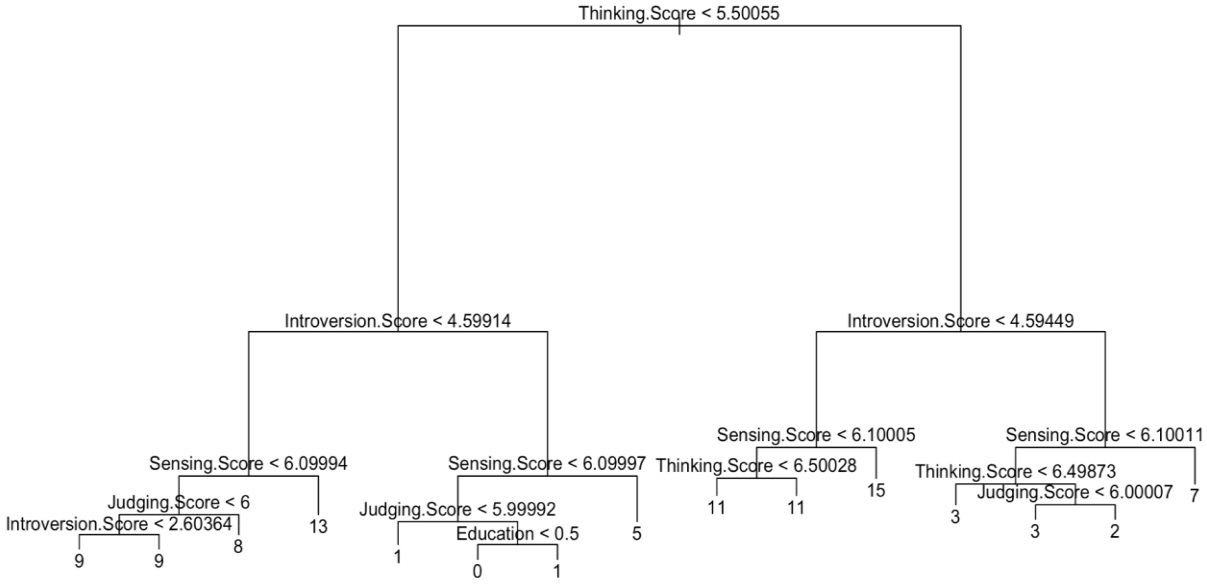
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.59	0.7	0.58	0.82	0.45	0.66	0.37	0.6	0.67	0.85	0.45	0.9	0.35	0.3	0.4	0.33



The graph above shows that the deviance increases as λ increases. A small λ should be selected as increasing λ might lead to underfitting. This indicates that the features are significantly relevant.

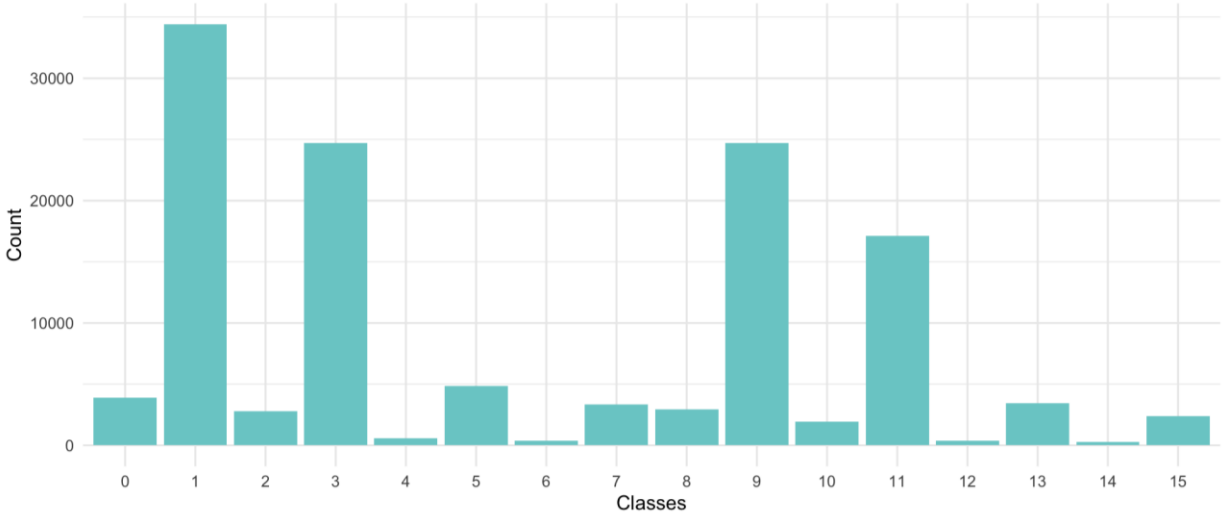
The optimal lambda for each fold is shown as follows. The average value of lambda is around 0.12, the model puts 0.12 penalty to the size of coefficients and shrinks the coefficients.

4. Decision Tree



Due to an imbalanced dataset, some classes are not in any of the leaf nodes. some classes are absent from the leaf nodes. To address this issue, we assigned weights to the classes with fewer observations.

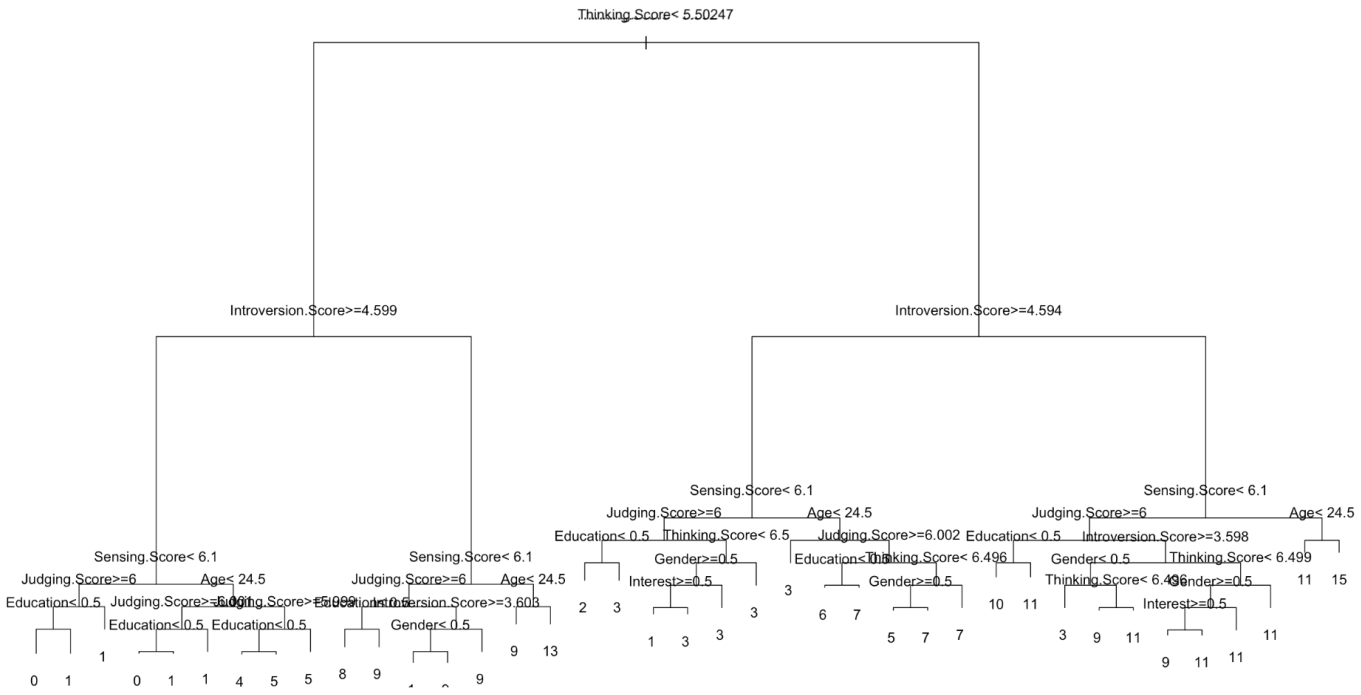
Classes Distribution



Based on the class distribution, we assigned a weight of 2 to classes {0, 2, 5, 7, 8, 10} and a weight of 4 to classes {4, 6, 12, 14}.

We also experimented with various hyperparameters using cross-validation. Different values for `minsplit`, `minbucket`, and `maxdepth` produced similar results. However, selecting a value of 0.001 for `cp` from the options {0.01, 0.001} led to improved model performance. A smaller cp allows for a more complex tree, better accommodating the minority classes.

With the above model tuning, the accuracy improved from 0.75 to 0.88; precision from 0.66 to 0.81; recall from 0.53 to 0.79.



Actual															
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	655	0	30	0	3	0	0	0	34	0	1	0	0	0	0	0
1	33	6273	0	113	2	45	0	0	6	326	0	4	0	0	0	0
2	0	0	384	0	0	0	1	0	0	0	14	0	0	0	0	0
3	0	69	80	4405	0	1	1	35	0	4	7	262	0	0	0	1
4	10	0	0	0	79	1	2	0	1	0	0	0	4	0	0	0
5	2	148	0	1	10	862	0	16	0	7	0	0	1	89	0	1
6	0	0	11	0	4	0	56	0	0	0	0	0	0	0	3	0
7	0	1	1	121	1	8	10	595	0	1	0	9	0	2	1	53
8	65	0	1	0	0	0	0	0	433	0	18	0	1	0	0	0
9	8	386	0	7	0	19	0	1	90	4498	0	61	3	87	0	3
10	0	0	40	0	0	0	0	0	0	0	271	0	0	0	0	0
11	0	2	5	289	0	0	0	1	0	43	51	3051	0	7	4	56
12	2	0	0	0	12	0	0	0	15	0	0	0	60	0	0	0
13	1	2	0	0	0	28	0	0	0	58	0	2	1	474	0	6
14	0	0	4	0	0	0	8	0	5	0	22	0	4	0	41	0
15	0	0	0	7	0	2	0	19	0	5	0	38	0	32	3	358

Actual															
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	635	0	16	0	3	0	0	0	21	0	0	0	0	0	0	0
1	50	6258	0	113	2	51	0	1	7	367	0	5	1	2	0	0
2	3	0	393	0	0	0	3	0	0	0	15	0	0	0	0	0
3	2	60	77	4435	0	5	0	53	0	2	3	260	0	0	0	0
4	15	0	0	0	73	1	0	0	0	0	0	0	5	0	0	0
5	0	125	0	7	16	868	0	12	0	14	0	0	4	96	0	1
6	2	0	10	0	6	0	62	0	0	0	0	0	1	0	2	0
7	0	1	1	88	0	8	8	575	0	0	0	3	1	7	1	65
8	58	0	4	0	2	0	0	0	464	0	14	0	5	0	0	0
9	10	428	0	17	0	11	0	1	85	4456	0	83	3	78	0	4
10	0	0	49	0	0	0	0	0	0	0	275	0	0	0	0	0
11	0	6	5	279	0	0	0	8	0	43	50	3030	0	9	1	64
12	2	0	0	0	8	0	0	0	6	0	0	0	38	0	1	0
13	0	3	0	0	0	23	0	0	0	59	0	0	11	456	0	8
14	0	0	2	0	0	0	6	0	1	0	27	0	5	0	42	0
15	0	0	0	5	0	0	0	16	0	1	0	46	0	43	6	336

Actual																
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	638	0	18	0	1	0	0	0	17	0	1	0	0	0	0	0
1	43	6240	0	120	1	52	0	0	9	333	0	9	1	1	0	0
2	1	0	393	0	0	0	0	0	1	0	10	0	0	0	0	0
3	1	70	66	4381	0	4	0	35	0	2	6	256	0	0	0	0
4	16	0	0	0	85	2	1	0	1	0	0	0	3	0	0	0
5	2	131	0	4	10	861	0	12	1	10	0	0	2	104	0	0
6	1	0	14	0	4	0	62	0	0	0	0	0	0	0	4	0
7	0	1	3	104	0	7	9	593	0	2	0	10	0	5	0	73
8	66	0	2	0	1	0	0	0	476	0	10	0	7	0	0	0
9	8	433	0	9	0	15	0	0	74	4464	0	73	0	87	0	1
10	0	0	50	0	0	0	0	0	0	0	276	0	0	0	0	0
11	0	2	7	324	0	0	0	10	2	63	53	3030	0	8	4	44
12	1	0	0	0	9	0	0	0	0	0	0	0	44	0	3	0
13	0	3	0	0	0	25	0	0	0	63	0	1	7	446	0	6
14	0	0	3	0	0	0	6	0	2	0	28	0	8	0	38	0
15	0	0	0	1	0	0	0	17	0	6	0	47	2	40	4	354

Actual																
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	637	0	16	0	1	0	0	0	20	0	1	0	0	0	0	0
1	40	6254	0	111	1	58	0	2	4	345	0	3	0	0	0	0
2	0	0	392	0	0	0	2	0	0	0	12	0	0	0	0	0
3	1	52	75	4414	0	7	0	38	0	5	5	232	0	0	0	1
4	19	0	0	0	91	0	0	0	0	0	0	0	5	0	0	0
5	3	130	0	1	10	853	0	11	0	8	0	0	2	78	0	0
6	1	0	15	0	1	0	63	0	0	0	1	0	0	0	2	0
7	0	1	3	86	0	4	6	584	0	0	0	8	0	8	1	74
8	70	0	1	0	0	0	0	0	470	0	9	0	2	0	1	0
9	5	433	0	14	0	15	0	0	76	4486	0	70	1	82	0	0
10	0	0	44	0	0	0	0	0	0	0	285	0	0	0	0	0
11	0	3	7	309	0	2	0	10	0	44	54	3072	1	8	1	64
12	1	0	0	0	7	0	0	0	8	0	0	0	48	0	0	0
13	0	8	0	0	0	28	0	0	2	54	0	0	8	472	0	8
14	0	0	4	0	0	0	7	0	4	0	17	0	7	0	40	0
15	0	0	0	9	0	0	0	22	0	0	0	41	0	43	7	331

Actual																
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	649	0	23	0	4	0	0	0	19	0	1	0	1	0	0	0
1	30	6285	0	117	0	56	0	1	9	352	0	9	0	2	0	0
2	3	0	391	0	0	0	2	0	0	0	11	0	0	0	0	0
3	0	40	68	4383	0	3	0	38	0	6	7	243	0	0	0	1
4	23	1	0	0	76	0	1	0	1	0	0	0	3	0	0	0
5	1	122	0	5	11	869	0	15	2	11	0	0	2	92	0	0
6	4	0	17	0	3	0	59	0	0	0	1	0	2	0	4	0
7	0	2	2	105	0	5	11	576	0	1	0	8	0	1	2	71
8	59	0	2	0	0	0	0	0	466	0	15	0	4	0	0	0
9	7	423	0	21	0	8	0	0	75	4456	0	65	1	84	0	1
10	0	0	48	0	0	0	0	0	0	0	273	1	0	0	0	0
11	0	8	4	311	0	1	0	13	0	45	55	3063	0	7	3	50
12	0	0	0	0	17	0	0	0	6	0	0	0	50	0	0	0
13	0	0	0	1	0	23	0	2	0	66	0	0	6	474	0	5
14	0	0	2	0	0	0	6	0	6	0	21	0	5	0	38	0
15	0	0	0	1	0	1	0	22	0	5	0	37	1	32	5	350

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Overall
Accuracy	0.88	0.87	0.87	0.88	0.88	0.88
Precision	0.81	0.81	0.81	0.82	0.79	0.81
Recall	0.80	0.78	0.79	0.80	0.79	0.79

Precision breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.93	0.92	0.96	0.91	0.78	0.76	0.73	0.74	0.85	0.87	0.86	0.86	0.71	0.82	0.49	0.75

Recall breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.83	0.91	0.70	0.89	0.73	0.89	0.77	0.88	0.79	0.90	0.72	0.89	0.65	0.67	0.76	0.72

5. Random Forest

Experimenting different number of predictors considered at each split (`mtry`) m with T = 500,

	m = p	m = p/2	m = √p
Accuracy	0.895	0.899	0.900
Precision	0.850	0.858	0.860
Recall	0.818	0.825	0.826

All models perform well with similar results; however, selecting m = √p yields the best outcome. Using a smaller m can reduce the risk of overfitting and obtain a more generalized model.

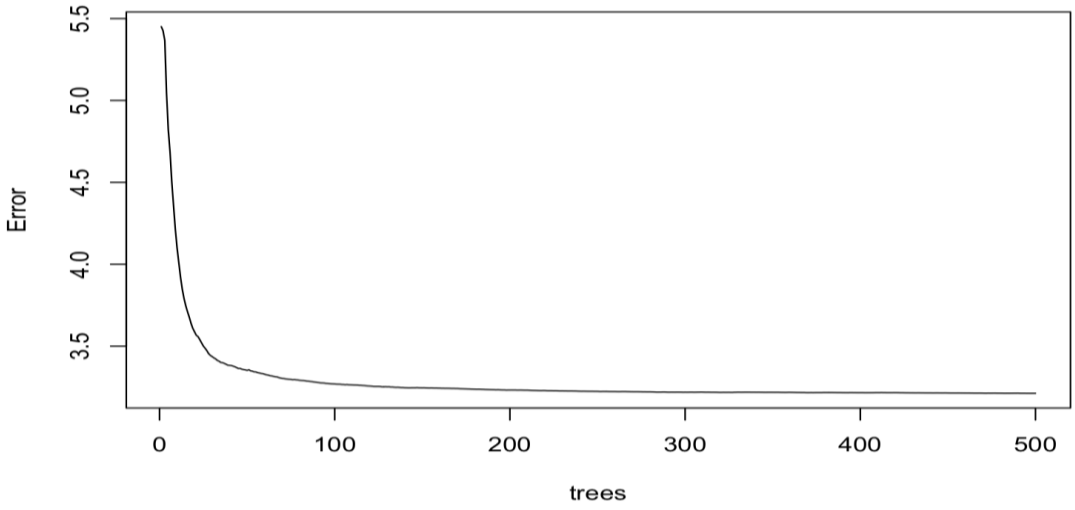
Experimenting different number of trees (`mtree`) T with m = √p,

	T = 100	T = 500
Accuracy	0.900	0.900
Precision	0.860	0.860
Recall	0.825	0.826

Using T=100 and T=500 produces similar results, with only a slight difference in recall, suggesting that the model captures data patterns effectively even with fewer trees. However, since T=500 shows marginally better performance, we will choose T=500.

Hyperparameters selection, m = √p; T=500.

Out-Of-Bag Error Rate



Confusion matrix:

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	704	0	0	0	11	0	0	0	48	0	0	0	0	0	0	0
1	7	6398	0	2	2	103	0	0	6	379	0	0	0	6	0	0
2	0	0	503	0	0	0	9	0	0	0	31	0	0	0	1	0
3	0	1	12	4588	0	0	1	57	0	0	3	320	0	0	0	2
4	8	0	0	0	79	0	2	0	2	0	0	0	7	0	0	0
5	2	89	0	0	9	793	0	5	0	5	0	0	1	47	0	0
6	0	0	7	0	0	0	51	0	0	0	1	0	0	0	4	0
7	0	0	2	74	0	0	8	564	0	0	0	2	0	0	0	27
8	53	0	0	0	0	0	0	0	503	0	0	0	3	0	0	0
9	2	391	0	1	0	10	0	1	14	4493	0	3	1	71	0	0
10	0	0	29	0	0	0	1	0	0	0	343	0	0	0	6	0
11	0	0	3	277	0	0	0	4	0	1	3	3051	0	0	2	62
12	0	0	0	0	9	0	0	0	11	0	0	0	59	0	0	0
13	0	2	0	0	1	59	0	1	0	64	0	1	3	567	0	5
14	0	0	0	0	0	0	6	0	0	0	3	0	0	0	32	0
15	0	0	0	1	0	1	0	35	0	0	0	50	0	0	7	382

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	699	0	2	0	7	0	0	0	40	0	0	0	1	0	0	0
1	22	6370	0	6	4	96	0	0	5	413	0	1	2	6	0	0
2	0	0	499	0	0	0	14	0	1	0	38	0	0	0	0	0
3	0	1	14	4610	0	0	0	76	0	0	2	302	0	0	0	4
4	9	0	0	0	84	0	0	0	1	0	0	0	5	0	0	0
5	0	86	0	0	9	808	0	3	0	5	0	0	3	50	0	0
6	0	0	4	0	0	0	53	0	0	0	0	0	0	0	5	0
7	0	0	1	53	0	0	8	542	0	0	0	2	0	0	0	27
8	37	0	2	0	1	0	0	0	518	0	1	0	8	0	0	0
9	8	417	0	0	0	7	0	0	15	4469	0	3	2	83	0	1
10	0	0	31	0	0	0	0	0	0	0	331	0	0	0	1	0
11	0	0	3	268	0	0	0	4	0	1	8	3081	0	0	1	63
12	2	0	0	0	5	0	0	0	4	0	0	0	41	0	1	0
13	0	7	0	0	0	56	0	0	0	54	0	0	12	552	0	6
14	0	0	1	0	0	0	4	0	0	0	4	0	0	0	38	0
15	0	0	0	7	0	0	0	41	0	0	0	38	0	0	7	377

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	700	0	0	0	5	0	1	0	48	0	0	0	1	0	0	0
1	16	6379	0	6	2	107	0	0	0	411	0	0	0	2	0	0
2	0	0	499	0	0	0	5	0	0	0	29	0	0	0	1	0
3	0	0	13	4589	0	0	0	83	0	0	4	274	0	0	0	7
4	14	0	0	0	94	0	0	0	0	0	0	0	15	0	0	0
5	1	79	0	0	3	786	0	1	0	3	0	0	0	41	0	0
6	0	0	7	0	0	0	60	0	0	0	0	0	0	0	4	0
7	0	0	1	45	0	0	5	547	0	0	0	2	0	0	0	37
8	43	0	0	0	1	0	0	0	504	0	0	0	3	0	1	0
9	2	416	0	0	0	9	0	0	21	4466	0	2	1	78	0	0
10	0	0	32	0	0	0	0	0	0	0	336	0	0	0	3	0
11	0	0	5	302	0	0	0	4	0	0	12	3105	0	0	1	55
12	0	0	0	0	3	0	0	0	8	0	0	0	43	0	1	0
13	1	7	0	0	3	65	0	0	3	62	0	1	11	570	0	5
14	0	0	0	0	0	0	6	0	0	0	3	0	0	0	34	0
15	0	0	0	2	0	0	1	32	0	0	0	42	0	0	7	374

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	702	0	1	0	8	0	0	0	41	0	0	0	0	0	0	0
1	13	6387	0	0	1	102	0	1	6	405	0	2	1	4	0	0
2	0	0	498	0	0	0	10	0	0	0	29	0	0	0	1	0
3	0	0	11	4561	0	0	0	68	0	0	5	331	0	0	1	5
4	12	0	0	0	88	0	1	0	0	0	0	0	6	0	0	0
5	1	82	0	1	6	801	0	2	1	9	0	0	1	57	0	0
6	0	0	6	0	0	0	55	0	0	0	1	0	0	0	6	0
7	0	0	3	58	0	0	8	553	0	0	0	3	0	0	0	44
8	41	0	1	0	2	0	0	0	530	0	1	0	12	0	0	0
9	8	411	0	0	0	10	0	0	4	4462	0	3	0	79	0	0
10	0	0	30	0	0	0	0	0	1	0	332	0	0	0	3	0
11	0	0	5	319	0	0	0	5	0	1	10	3027	0	0	3	37
12	0	0	0	0	5	0	0	0	0	0	0	0	44	0	3	0
13	0	0	0	0	1	53	0	0	0	66	0	0	10	551	0	1
14	0	0	1	0	0	0	4	0	0	0	6	0	0	0	33	0
15	0	0	0	4	0	0	0	38	0	0	0	60	0	0	3	391

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	709	0	1	0	9	0	0	0	41	0	0	0	1	0	0	0
1	3	6398	0	5	1	121	0	1	3	430	0	0	0	8	0	0
2	0	0	511	0	0	0	2	0	0	0	37	0	0	0	3	0
3	0	0	9	4554	0	0	0	70	0	0	4	281	0	0	1	5
4	15	0	0	0	80	0	1	0	2	0	0	0	10	0	0	0
5	2	76	0	0	8	796	0	1	0	4	0	0	1	60	0	0
6	1	0	8	0	0	0	60	0	0	0	0	0	0	0	3	0
7	0	0	1	54	0	0	11	543	0	0	0	4	0	0	1	31
8	44	0	0	0	2	0	0	0	527	0	3	0	10	0	0	0
9	2	404	0	3	0	3	0	0	6	4444	0	3	1	74	0	0
10	0	0	25	0	0	0	0	0	0	0	324	0	0	0	3	0
11	0	1	2	322	0	0	0	7	0	0	11	3100	0	0	2	42
12	0	0	0	0	11	0	1	0	4	0	1	0	43	0	0	0
13	0	2	0	0	0	46	0	1	1	64	0	0	8	550	0	2
14	0	0	0	0	0	0	4	0	0	0	4	0	1	0	34	0
15	0	0	0	6	0	0	0	44	0	0	0	38	0	0	5	398

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Overall
Accuracy	0.90	0.90	0.90	0.90	0.90	0.90
Precision	0.81	0.81	0.81	0.81	0.81	0.81
Recall	0.79	0.79	0.79	0.79	0.79	0.79

Precision breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.93	0.92	0.92	0.92	0.80	0.84	0.83	0.84	0.91	0.90	0.91	0.89	0.77	0.80	0.78	0.80

Recall breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.90	0.93	0.90	0.93	0.77	0.82	0.71	0.82	0.88	0.90	0.87	0.90	0.62	0.81	0.65	0.80

6. Support Vector Machine

(linear kernel)

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	640	27	12	1	37	3	0	0	36	1	0	0	1	0	0	0
1	106	6078	0	130	2	324	0	5	3	406	0	8	0	16	0	0
2	15	0	434	15	1	0	32	0	1	0	17	2	0	0	2	0
3	2	128	86	4361	0	8	2	255	0	5	6	281	0	2	0	22
4	25	0	0	0	66	5	1	0	2	0	0	0	5	0	0	0
5	2	197	0	1	11	559	0	8	0	14	0	0	2	33	0	0
6	1	0	8	1	1	0	50	0	0	0	0	0	0	0	1	0
7	0	2	2	114	0	10	5	372	0	0	0	7	0	1	1	25
8	35	4	1	0	6	2	0	0	424	25	11	4	29	1	3	0
9	6	364	0	7	0	22	0	1	49	4248	0	66	4	249	0	1
10	2	0	26	5	0	0	3	0	7	0	281	24	0	0	15	0
11	0	10	5	281	0	1	0	12	10	97	47	2932	0	7	0	161
12	0	0	0	0	2	0	0	0	8	0	0	0	29	1	0	0
13	0	13	0	0	2	31	0	0	3	146	0	1	10	391	0	5
14	0	0	0	0	0	0	3	1	0	0	6	0	1	0	25	2
15	0	1	0	10	0	0	0	18	0	3	0	83	0	12	8	265

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	596	45	8	1	36	4	0	0	37	0	0	0	5	0	0	0
1	107	6124	0	112	6	301	0	4	10	374	0	5	1	22	0	0
2	18	1	425	30	1	0	15	2	0	0	37	1	0	0	5	0
3	6	141	52	4425	0	13	7	251	1	5	6	312	0	3	0	22
4	26	0	0	0	39	4	0	0	2	0	0	0	6	0	0	0
5	1	190	0	5	4	578	0	10	0	11	0	0	0	25	0	0
6	0	0	14	0	1	0	42	2	0	0	2	0	0	0	2	0
7	1	4	2	117	0	15	1	360	0	0	0	5	0	2	0	21
8	35	4	0	0	3	0	0	0	399	34	10	1	26	1	1	0
9	3	330	0	7	0	13	0	0	80	4345	0	80	2	233	0	4
10	2	0	22	2	0	0	0	0	10	0	291	14	0	0	7	0
11	1	11	5	263	0	0	0	13	2	93	53	2852	0	2	1	157
12	1	0	0	0	4	0	0	0	13	0	1	0	31	0	0	0
13	1	9	0	0	0	37	0	1	1	129	0	1	9	394	0	4
14	0	0	1	0	0	0	1	1	1	0	6	0	0	0	23	3
15	0	0	0	10	0	0	0	32	0	0	0	112	0	11	1	293

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	547	38	9	1	37	4	2	0	40	5	2	0	1	0	0	0
1	114	6212	0	107	4	303	0	7	8	373	0	5	1	21	0	0
2	4	0	394	17	0	0	28	1	0	0	27	3	0	0	2	0
3	3	118	84	4414	0	11	2	248	0	11	8	275	0	1	1	15
4	17	0	0	0	59	4	0	0	1	0	0	0	5	0	0	0
5	5	190	0	1	9	589	0	18	0	11	0	0	0	39	0	0
6	1	0	5	0	1	0	34	0	0	0	1	0	0	0	1	0
7	0	3	2	141	2	7	7	358	0	0	1	10	0	1	4	18
8	36	5	2	0	2	0	0	0	439	36	5	1	24	1	0	0
9	3	342	0	5	0	16	0	0	62	4299	0	92	2	238	0	6
10	1	0	36	2	0	0	0	0	11	0	289	24	0	0	14	1
11	1	10	6	268	0	0	0	7	5	82	50	2929	0	2	0	151
12	0	0	0	0	4	1	0	0	10	1	0	0	31	1	0	0
13	0	6	0	0	0	28	0	1	1	129	0	2	3	371	0	5
14	0	0	2	0	0	0	2	1	0	0	10	0	5	0	25	0
15	1	0	0	4	0	2	0	25	0	0	1	91	0	9	10	271

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	566	43	16	3	31	0	1	0	43	6	0	0	2	0	0	0
1	112	6094	0	123	2	294	0	8	9	349	0	6	1	19	0	2
2	10	0	436	22	0	0	28	1	0	1	24	2	0	0	4	1
3	5	135	62	4349	0	6	1	244	0	6	3	257	0	0	0	10
4	21	0	0	0	55	2	0	0	3	0	0	0	3	0	0	0
5	4	216	0	1	6	578	0	4	2	19	0	0	1	33	0	0
6	1	0	15	2	0	0	37	1	0	0	0	0	0	0	4	0
7	1	3	5	150	1	6	3	374	0	1	0	11	0	0	0	30
8	48	4	0	0	3	0	0	0	469	30	9	1	12	1	1	0
9	8	375	0	7	0	35	0	1	69	4284	0	81	1	229	0	6
10	0	0	20	2	0	0	0	0	7	1	283	20	0	0	14	0
11	1	3	3	273	0	1	0	10	6	86	52	2952	0	6	3	166
12	1	0	0	0	5	0	0	0	16	0	0	0	38	0	0	0
13	0	7	0	0	0	37	0	0	2	134	0	4	6	409	0	11
14	0	0	0	0	0	0	5	0	0	0	14	0	0	0	23	2
15	0	0	0	11	0	1	0	23	0	2	0	103	0	5	6	235

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	556	36	10	2	32	0	1	0	37	3	0	0	5	0	0	0
1	106	6122	0	118	6	283	0	4	5	353	0	10	2	24	0	0
2	10	0	449	17	3	0	17	0	0	0	25	2	0	0	2	0
3	5	138	74	4364	0	6	5	214	0	4	5	263	0	0	2	13
4	22	0	1	0	58	6	1	1	0	0	0	0	2	0	0	0
5	2	198	0	3	5	607	0	4	1	10	0	1	2	32	0	0
6	1	0	15	0	1	0	39	0	0	0	1	0	0	0	3	0
7	0	4	2	154	1	16	13	382	0	0	0	8	0	0	0	24
8	31	0	1	0	2	0	0	0	470	29	12	0	16	2	0	0
9	4	396	0	4	0	17	0	0	76	4286	0	78	5	204	0	7
10	0	0	25	2	0	0	0	1	9	0	269	12	0	0	14	1
11	1	6	5	244	0	3	0	16	4	92	48	3006	1	6	7	154
12	2	1	0	0	3	0	1	0	10	1	0	0	28	1	0	0
13	0	16	0	1	0	37	0	1	3	131	0	3	11	387	0	6
14	0	0	1	0	0	0	3	1	2	0	7	0	1	0	22	3
15	0	0	0	8	0	2	0	30	0	0	0	89	1	8	5	267

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Overall
Accuracy	0.82	0.82	0.83	0.82	0.83	0.83
Precision	0.76	0.73	0.74	0.72	0.73	0.74
Recall	0.66	0.67	0.66	0.67	0.66	0.67

Precision breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.82	0.76	0.82	0.63	0.75	0.68	0.77	0.73	0.75	0.71	0.80	0.59	0.71	0.65	0.72	0.75

Recall breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.75	0.89	0.77	0.89	0.5	0.6	0.52	0.55	0.75	0.77	0.72	0.71	0.70	0.75	0.45	0.55

(with polynomial kernel)

	Actual														
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	643	45	7	0	29	2	2	0	40	2	0	0	0	0	0	0
1	121	6279	5	66	3	220	0	3	6	432	0	9	1	16	0	0
2	10	0	446	30	0	0	22	1	0	0	14	4	0	0	2	0
3	0	38	81	4490	0	0	1	151	0	2	10	302	0	1	0	13
4	10	0	0	0	76	4	0	0	1	0	0	0	5	0	0	0
5	1	64	0	0	10	678	0	6	0	7	0	0	1	45	0	0
6	0	0	5	0	1	0	54	3	0	0	0	0	0	0	1	0
7	0	0	3	38	0	8	11	466	0	0	0	4	0	0	1	40
8	44	2	0	0	3	0	1	0	415	28	2	0	23	2	1	0
9	4	388	0	4	0	14	0	0	73	4383	4	35	4	187	0	2
10	1	0	22	5	0	0	1	0	5	1	278	28	0	0	13	0
11	0	4	5	291	0	0	1	13	0	38	57	2987	0	2	0	83
12	0	0	0	0	3	1	0	0	3	0	0	0	37	2	1	0
13	0	4	0	0	3	38	0	0	0	51	0	0	10	453	1	2
14	0	0	0	0	0	0	3	1	0	0	3	1	0	0	27	2
15	0	0	0	2	0	0	0	28	0	1	0	38	0	5	8	339

	Actual														
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	609	38	5	0	25	0	0	0	36	4	1	0	5	0	0	0
1	130	6348	3	70	5	199	0	2	18	406	1	3	1	23	0	0
2	11	1	419	37	0	0	13	2	0	0	38	1	0	0	3	0
3	2	42	66	4534	0	1	4	134	0	3	8	330	0	1	0	12
4	6	1	0	0	51	5	1	0	0	0	0	0	6	0	0	0
5	0	71	0	0	4	702	0	9	0	5	0	0	1	35	0	1
6	0	0	8	0	1	0	46	5	0	0	0	0	0	0	5	0
7	0	1	0	55	0	12	2	479	0	0	0	3	0	0	0	33
8	33	0	0	0	1	0	0	0	403	29	11	0	12	0	0	0
9	5	352	0	5	0	7	0	0	87	4470	2	39	5	148	0	2
10	1	0	21	2	0	0	0	0	7	0	276	18	0	0	6	0
11	0	4	6	265	0	0	0	9	0	34	60	2944	0	2	0	96
12	0	0	0	0	6	0	0	0	4	0	0	0	37	6	2	0
13	1	1	0	1	1	39	0	0	1	40	1	0	13	473	0	4
14	0	0	1	0	0	0	0	1	0	0	8	0	0	0	22	1
15	0	0	0	3	0	0	0	35	0	0	0	45	0	5	2	355

	Actual														
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	587	42	13	0	18	1	1	0	44	4	0	0	1	0	0	0
1	125	6323	4	67	2	212	0	0	12	395	0	3	2	19	0	0
2	5	1	428	31	0	0	19	1	0	1	25	1	0	0	3	0
3	0	50	81	4508	0	1	0	160	0	4	3	283	0	0	0	7
4	5	0	0	0	62	4	0	0	0	0	0	0	5	0	0	0
5	0	58	0	0	14	660	0	9	1	5	0	0	2	39	0	0
6	0	0	3	1	0	0	45	3	0	0	0	0	0	0	3	0
7	0	0	2	51	0	5	4	459	0	0	0	7	0	0	2	32
8	44	3	0	0	2	0	0	0	476	29	5	0	9	0	1	0
9	12	401	0	5	0	34	0	0	78	4387	1	31	1	128	0	2
10	0	0	21	0	0	0	1	0	6	1	289	24	0	0	12	1
11	0	1	5	277	0	1	0	8	0	37	58	3050	0	6	4	121
12	0	0	0	0	5	0	0	0	8	0	0	0	38	3	1	0
13	0	1	0	0	0	42	0	1	1	56	0	0	6	505	0	6
14	0	0	0	0	0	0	5	0	0	0	4	1	0	0	25	3
15	0	0	0	3	0	0	0	25	0	0	0	37	0	2	4	291

Actual															
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	541	36	3	1	31	2	1	0	39	2	1	0	2	0	0	0
1	139	6421	5	52	5	199	0	3	15	409	0	2	2	19	0	0
2	1	0	402	33	0	0	19	0	1	0	27	0	0	0	3	0
3	0	35	83	4540	0	3	0	149	0	3	12	305	0	0	2	14
4	4	1	0	0	66	6	0	0	0	0	0	0	5	0	0	0
5	2	66	0	0	8	706	1	7	0	8	0	0	0	49	0	0
6	1	0	2	0	0	0	40	2	0	0	0	0	0	0	1	0
7	0	1	1	59	0	7	12	471	0	0	0	3	0	0	3	26
8	39	3	1	0	2	0	0	0	435	37	7	0	18	1	0	0
9	4	354	1	8	0	10	0	0	73	4420	0	42	2	144	0	3
10	2	0	34	2	0	0	0	0	6	1	288	28	0	0	8	0
11	0	4	8	265	0	0	0	5	0	26	55	3019	0	2	2	80
12	0	0	0	0	5	0	0	0	7	0	0	0	32	5	0	0
13	0	3	0	0	0	30	0	1	1	41	0	1	7	459	0	4
14	0	0	0	0	1	0	2	0	0	0	4	0	4	0	30	4
15	0	0	0	0	0	2	0	28	0	0	0	32	0	5	8	336

Actual															
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	563	44	6	0	24	2	0	0	46	0	0	0	4	0	0	0
1	125	6343	3	64	5	184	0	1	12	390	0	7	2	17	0	0
2	4	0	439	31	0	0	12	1	0	0	28	1	0	0	2	0
3	0	46	85	4494	0	1	2	136	0	1	5	290	0	0	2	9
4	8	0	0	0	62	6	2	0	0	0	0	0	6	0	0	0
5	1	75	0	1	10	710	0	6	0	2	0	1	1	43	0	0
6	0	0	7	0	2	0	43	1	0	0	1	0	0	0	3	0
7	0	1	1	55	1	13	17	458	0	0	0	2	0	0	0	29
8	34	2	1	0	0	0	0	0	458	32	7	0	11	1	1	0
9	5	400	0	1	0	11	0	0	93	4403	3	31	3	131	0	2
10	0	0	36	1	0	0	1	0	4	1	274	23	1	0	9	1
11	0	2	5	268	0	2	0	14	0	33	44	3085	0	0	5	97
12	0	0	0	0	6	0	1	0	3	0	0	0	30	1	0	0
13	0	4	0	0	1	47	0	0	0	47	0	1	15	463	1	4
14	0	0	0	0	0	0	2	1	1	0	5	0	1	0	26	3
15	0	0	0	2	0	1	0	36	0	0	0	31	0	8	6	330

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Overall
Accuracy	0.86	0.87	0.87	0.86	0.87	0.86
Precision	0.82	0.80	0.81	0.82	0.81	0.81
Recall	0.72	0.72	0.72	0.73	0.71	0.72

Precision breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.83	0.90	0.83	0.90	0.69	0.82	0.74	0.79	0.80	0.82	0.78	0.87	0.65	0.78	0.71	0.79

Recall breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.76	0.90	0.77	0.90	0.55	0.76	0.60	0.73	0.77	0.90	0.75	0.87	0.43	0.74	0.47	0.73

(with radial kernel)

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	643	64	4	1	26	3	0	0	40	2	0	0	0	0	0	0
1	115	6227	4	48	1	167	0	0	5	423	0	5	0	11	0	0
2	10	2	445	41	0	0	22	1	0	0	18	6	0	0	2	0
3	0	52	79	4471	0	0	2	116	0	1	4	290	0	1	0	3
4	14	0	0	0	72	4	1	0	1	0	0	0	5	1	0	0
5	0	75	0	0	17	726	0	9	0	10	0	0	2	43	0	0
6	1	0	6	0	1	0	55	4	0	0	0	0	0	0	2	0
7	0	1	3	51	0	10	10	497	0	0	0	3	0	1	1	45
8	45	3	0	0	3	0	0	0	425	39	4	2	23	1	2	0
9	5	393	0	3	1	13	0	0	65	4381	0	34	4	146	0	2
10	1	0	27	5	0	0	1	0	4	1	281	26	0	0	11	0
11	0	4	6	303	0	0	1	11	0	32	58	3000	0	4	0	73
12	0	0	0	0	5	2	0	0	3	0	0	0	38	2	0	0
13	0	3	0	0	2	39	0	0	0	54	0	1	9	497	0	0
14	0	0	0	0	0	0	4	1	0	0	3	1	0	0	27	2
15	0	0	0	3	0	1	0	33	0	2	0	40	0	6	10	356

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	607	55	5	0	23	0	0	0	35	7	2	0	2	0	0	0
1	131	6313	2	45	2	155	0	1	14	398	0	2	0	8	0	0
2	11	4	429	48	0	0	12	0	0	0	37	2	0	0	3	0
3	1	38	56	4528	0	1	2	98	0	2	9	318	0	1	0	9
4	7	3	0	0	55	5	0	0	0	0	0	0	8	1	1	0
5	0	75	0	0	6	737	0	12	0	5	0	0	2	45	0	0
6	0	0	5	0	1	0	47	6	0	0	0	0	0	0	5	0
7	0	4	4	62	0	13	5	510	0	0	0	1	0	1	0	31
8	34	1	0	0	1	0	0	0	408	37	7	1	15	0	0	0
9	6	359	0	4	0	5	0	0	84	4467	3	39	5	125	0	0
10	0	0	21	2	0	0	0	0	7	0	286	17	0	0	6	0
11	0	6	6	280	0	0	0	4	0	30	57	2953	0	1	0	87
12	0	0	0	0	5	0	0	0	6	0	0	0	37	1	1	0
13	1	1	0	0	1	49	0	0	2	44	1	0	11	505	0	4
14	0	0	1	0	0	0	0	0	0	0	4	0	0	0	21	2
15	0	0	0	3	0	0	0	45	0	1	0	50	0	5	3	371

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	546	46	4	2	22	2	1	0	39	5	1	0	1	0	0	0
1	128	6368	1	48	2	132	0	1	9	402	0	5	1	12	0	0
2	3	0	409	42	0	0	19	2	1	0	33	0	0	0	1	0
3	1	45	79	4524	0	2	0	99	0	2	8	287	0	0	1	7
4	6	2	0	0	73	5	0	0	0	0	0	0	5	0	0	0
5	3	77	0	1	13	772	1	14	1	5	0	0	0	52	0	0
6	1	0	1	0	0	0	41	2	0	0	0	0	0	0	2	0
7	0	0	3	56	0	6	10	512	0	0	0	4	0	1	3	26
8	40	8	1	0	2	0	0	0	441	46	4	1	19	1	0	0
9	4	371	0	8	0	7	0	0	73	4403	1	47	1	102	0	1
10	1	0	35	2	0	0	0	0	5	0	286	35	0	0	8	0
11	0	4	7	275	0	0	0	5	1	36	56	3014	0	2	1	66
12	0	0	0	0	6	0	0	0	7	0	0	0	35	5	0	0
13	0	3	0	0	0	37	0	0	0	48	0	2	8	503	0	2
14	0	0	0	0	0	0	3	0	0	0	5	0	2	0	33	5
15	0	0	0	2	0	2	0	31	0	0	0	37	0	6	8	360

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	580	60	12	1	15	2	1	0	47	5	0	0	1	0	0	0
1	122	6267	3	43	3	155	0	3	10	364	0	3	1	12	0	0
2	6	1	428	47	0	0	22	2	0	1	24	4	0	0	3	0
3	1	57	80	4488	0	0	0	129	0	3	4	278	0	0	0	6
4	10	0	0	0	62	3	0	0	0	0	0	0	4	0	0	0
5	1	72	0	0	14	714	0	10	2	10	0	0	2	48	0	1
6	0	0	5	0	0	0	42	5	0	0	0	0	0	0	3	0
7	0	0	1	55	0	3	5	478	0	0	0	5	0	0	2	35
8	49	5	0	0	2	0	0	0	479	44	6	0	9	1	1	0
9	9	413	0	2	0	25	0	0	73	4377	1	33	2	92	0	1
10	0	0	20	1	0	0	0	0	6	1	290	24	0	0	11	1
11	0	2	8	300	0	0	0	5	0	39	55	3046	0	3	5	102
12	0	0	0	0	7	0	0	0	6	0	0	0	35	3	2	0
13	0	3	0	0	0	57	0	1	3	75	0	0	10	538	0	8
14	0	0	0	0	0	0	5	0	0	0	4	1	0	0	23	3
15	0	0	0	6	0	1	0	33	0	0	1	43	0	5	5	306

Predicted \ Actual	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	563	48	5	0	21	1	0	0	40	1	0	0	2	0	0	0
1	117	6316	3	43	4	151	0	0	9	371	0	4	3	9	0	0
2	4	1	446	36	0	0	10	1	0	0	28	3	0	0	2	0
3	0	43	81	4491	0	1	1	96	0	3	3	277	0	0	2	6
4	13	0	0	0	61	7	0	0	0	0	0	0	7	0	0	0
5	1	78	0	1	14	738	1	9	0	4	0	0	1	47	0	0
6	0	0	7	0	3	0	48	1	0	0	1	0	0	0	3	0
7	0	1	1	65	0	13	16	493	0	0	0	4	0	0	0	30
8	38	3	2	0	0	0	0	0	475	42	8	0	9	1	1	0
9	4	419	0	1	0	8	0	0	85	4400	3	41	1	104	0	1
10	0	0	33	0	0	0	0	1	5	1	276	26	1	0	11	1
11	0	2	5	277	0	0	0	12	0	34	44	3079	0	0	0	82
12	0	0	0	0	7	1	1	0	2	0	0	0	32	5	1	0
13	0	6	0	0	1	54	0	0	1	53	0	3	17	492	0	5
14	0	0	0	0	0	0	3	0	0	0	4	1	1	0	25	1
15	0	0	0	3	0	3	0	41	0	0	0	34	0	6	10	349

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Overall
Accuracy	0.86	0.87	0.87	0.86	0.87	0.87
Precision	0.81	0.81	0.81	0.79	0.80	0.81
Recall	0.73	0.74	0.74	0.73	0.73	0.73

Precision breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.83	0.90	0.84	0.90	0.69	0.83	0.74	0.79	0.82	0.87	0.78	0.87	0.65	0.78	0.71	0.78

Recall breakdown for each class:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.76	0.91	0.77	0.91	0.55	0.76	0.60	0.75	0.77	0.90	0.75	0.87	0.43	0.74	0.47	0.73

Challenges

1. Method Choice

   We attempted to apply QDA, but encountered errors when running it.

```
QDA failed for fold 1
QDA failed for fold 2
QDA failed for fold 3
QDA failed for fold 4
QDA failed for fold 5
```

   Here are some potential reasons for the errors:

   - There are some classes that have very few instances compared to others, it can affect the stability of the covariance estimates and lead to errors.
   - Most of the variables are categorical while QDA assumes continuous variables.

   Since other methods that we tried are performing well, we will focus on other methods.

2. Imbalance dataset

   As we can see from the result, the classes with fewer observations are generally performing poorer. In Ridge Regression, some of these classes have no observations predicted at all.

   E.g. class 4, 6, 10, 12, 14:

	Actual															
Predicted	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	73	2	15	0	24	0	1	0	20	0	2	0	5	0	0	0
1	613	6361	7	157	55	707	1	8	38	400	0	7	4	38	0	0
2	0	0	19	0	0	0	4	0	0	0	2	0	0	0	1	0
3	11	117	508	4504	3	13	48	522	0	7	30	313	0	1	5	28
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	7	33	1	4	23	201	2	35	1	4	0	1	6	33	0	3
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	5	6	0	0	21	51	0	0	1	0	0	0	2	19
8	0	0	0	0	1	0	1	0	22	0	9	0	5	0	1	0
9	34	395	1	8	4	45	0	0	523	4428	11	115	42	507	0	8
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	1	7	27	238	0	5	0	37	9	58	312	3031	2	12	40	402
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	1	2	0	0	1	6	1	1	4	12	0	3	10	73	2	9
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	2	0	0	4	6

   To deal with this issue, we use more complex model that are capable for handling imbalance dataset like Random Forest and SVM. We also tried tuning the model with hyperparameters, and in Decision Trees, we saw a significant improvement when we added weights to the classes and used different hyperparameters.

Summary

Important dimensions and why they are important

Features	**Importance** (Mean Decrease Impurity)
Introversion Score	614.6
Sensing Score	420.0
Thinking Score	337.2
Gender	237.2
Age	200.4
Judging Score	37.5
Interest	34.3
Education	27.7

We obtained the above importance from the best performing model, Random Forest, by calculating the mean decrease impurity. Introversion score is the most significant, and judging score, interest, education might not be significant as others.

Learnings from the result

Method	Success Metrics		
	Accuracy	Precision	Recall
Logistic Regression	89.7 %	71.2 %	62.2 %
Logistic Regression (PCA)	86.9 %	60.3 %	60.4 %
Lasso Regression	82.7 %	72.9 %	66.2 %
Ridge Regression	73.1 %	65.0 %	26.2 %
Decision Tree	87.6 %	80.9 %	79.4 %
Random Forest	90.4 %	81.3%	79.5 %
Support Vector Machine (linear)	82.9 %	74.0 %	66.8 %
Support Vector Machine (polynomial kernel)	86.5 %	81.2 %	72.1 %
Support Vector Machine (radial kernel)	86.8 %	80.8 %	73.5 %

The above table compares the metrics for all the methods that we have used for obtaining the MBTI model. Following learning outcomes can be drawn from these results:

- The linear models have overall very less recall and precision which signifies that the true positives are not efficiently captured for each class. The logistic model outperforms ridge and lasso in terms of accuracy while lasso outperforms in terms of recall and precision due to its inability to shrink irrelevant coefficients.
- We also observed that dimension selectivity by using PCA is not very useful since the results obtained by scaling the parameters are better than results obtained using PCA.
- The results obtained from the decision tree are moderate which suggests that it is interpretable but it cannot capture the complex relationship among the parameters accurately.
- The SVM methods performed well in terms of accuracy but we can observe that the non-linear methods (polynomial and kernel) yield better recall and precision than the linear method. This is inline with what we observed for the linear models.
- It is evident from the results that the random forest has the highest values among all the methods for all the metrics (highlighted values in the above table). It clearly suggests that this model efficiently captures the non-linearity and the

interactions among the features. This might be due to the ensemble approach that random forest uses to predict the output from several decision trees, reducing overfitting and making a more generic model.

Future scope

To enhance the MBTI personality prediction we can incorporate more personality and demographic traits like Ethnicity, Location, Profession, etc to make a more robust and complex prediction. We can use enhanced preprocessing techniques like Natural Language Processing (NLP) and word embedding to analyze the text features and use them in prediction. We can also combine methods like Random Forests and SVM with techniques like hyperparameter optimization to further improve the prediction of models. Also, using an extensive dataset with a high number of observations can yield a very good result since we have a multi-class prediction with many features, however, the computational power is a big limitation in achieving this goal which can be mitigated in future for expanding the scope of this project.