CSC311 Machine Learning Challenge Report

David Chang, Aaron Liu, Neil Mehta, Vishnu Ravinder

Introduction

The goal of this lab report is to communicate the method through which this group developed a classifier using the provided data set to predict which city a student is referring to in their response to the quiz conducted earlier in the semester by instructors.

In this lab report, the key concepts covered include the sorting and handling of the provided dataset, the models considered and tested, implementation considerations (ie., hyperparameters, evaluation metrics, etc.), a description of the final model implementation, and expected prediction performance.

During this challenge, Neil was responsible for writing the lab report. Vishnu was responsible for examing the MLP model. Aaron was responsible for examining the KNN model. David was responsible for examining the Decision Tree Model.

Data Cleaning and Sorting

Handling Empty Fields

One approach to handling empty fields in the dataset that we considered was to remove any rows that contained empty fields. While this was a simpler solution, it risked the loss of valuable data and might have potentially reduced the performance of the model on the test set. Alternatively, another approach considered was to fill empty fields in with either the mean, median, or mode of the non-empty values for that field. Although this approach required more effort, it ensured that more data was retained. However, for question 6, which involved ranking different items from 1 to 6, filling in empty values was challenging, and no suitable method was found to address this issue.

Converting Values into Numbers

To make the dataset more amenable to calculations, the values for questions Q5 and Q6 were converted into separate columns for each part of the question. For example, for Q6, each item that needed to be ranked (skyscrapers, sports, arts and music, carnival, cuisine, economics) was transformed into separate columns, and the corresponding rank was assigned. This conversion ensured that all data were represented as numbers (excluding Q10).

Cleaning Text

The Text in Q10 was completely generated by the user, as such it contained a lot of characters and words that were useless for analysis. To handle odd characters we decided to replace any character that wasn't alphanumeric or a space. We knew that we would eventually be converting the text into a bag of words, so we wanted to be able to easily split on spaces and not have to worry about "hello" being considered different from "hello,". This also helped remove any odd characters such as non readable characters, newlines, or tabs that would other muck up our model.

Pruning Values

To ensure that the data was within sensible ranges, the values in various fields were pruned. For example, the temperature in the photo (Q7) was capped at 45 and had a minimum of -20. Which is slightly higher then the hottest recorded temperature of Rio without factoring in humidity. And equal to the lowest temperature of every experienced in Paris. Similarly, the number of different languages spoken (Q8) was limited to a maximum of 10 and a minimum of 1. 1 is an obvious choice for the minimum since at least one language must be spoken, and we set a max of 10 after consulting many online forums where people gave numbers that were quite a bit lower then our cutoff. For Q9 (different fashion styles), the values ranged from 1 to 10. Once again we need to have atleast one fashion style no matter where you are and 10 as a max because people aren't that creative lets be honest. Additionally, for Q6, where rankings should be distinct, any rows where the rankings were not unique were removed to maintain data integrity. Similarly blank rows in Q10

or rows that were made blank after cleaning the text would also have to be removed as we didn't know a way to fill those rows with sample text.

After deleting some of the rows there was in balance in the number of rows with each label. This could cause bias towards one label so we remove some rows till each label has an equal amount.

Splitting into Different Sets

For the task of splitting the data into different sets, k-fold cross-validation was employed, with k = 5 to construct a more robust model with a less optimistic estimate of the test accuracy. The dataset was initially divided into 4 groups based on the label to ensure an equal distribution of each label in every fold. This approach helped prevent bias in the model and provided a more reliable evaluation of its performance. We then iterate though the various folds with every fold being designated the test set once. The other four are designated the training set and of that training set 20% is taken and designated as the validation set for hyper parameter tuning.

Feature Analysis

To gain insights into the dataset, we conducted feature analysis using various techniques. Firstly, we utilized a correlation map to examine the relationships between features. The goal was to identify highly correlated features, as such redundancy could simplify the model and reduce computation time. While the Q6 columns exhibited strong correlation, as expected due to the nature of the ranking, no other features showed excessively high correlation, indicating the importance of retaining all features:

	id	Q1	Q2	Q3	Q4	Q7	Q8	Q9	Q5_Partner	Q5_Friends	Q5_Siblings	Q5_Co-worker	Q6_Skyscraper	Q6_Sport	Q6_Art_and_Music	Q6_Carnival	Q6_Cuisine	Q6_Economics
id	1.000000	-0.007946	-0.021002	-0.053404	0.005924	0.002485	0.021471	0.001116	-0.060875	0.005929	0.045925	-0.031605	-0.007263	-0.012333	0.017180	-0.000061	-0.045108	-0.009231
Q1	-0.007946	1.000000	0.441867	0.300584	-0.063146	-0.400556	0.107360	0.243594	0.165788	-0.024624	0.128528	0.126227	0.251940	-0.277034	0.049725	-0.255610	0.174506	0.306081
Q2	-0.021002	0.441867	1.000000	0.188465	0.136769	-0.329200	0.220369	0.300727	0.034619	0.099752	0.127167	0.203966	0.288602	-0.153459	-0.029904	-0.145800	0.012567	0.272768
Q3	-0.053404	0.300584	0.188465	1.000000	0.043624	-0.091893	0.051826	0.133615	0.158118	-0.001680	0.058572	-0.058218	0.152946	-0.150643	0.056070	-0.119064	0.192120	0.145907
Q4	0.005924	-0.063146	0.136769	0.043624	1.000000	0.187793	-0.002173	0.051743	0.048369	0.178767	-0.014079	-0.125714	-0.264792	0.330094	0.231753	0.490247	-0.038573	-0.253690
Q7	0.002485	-0.400556	-0.329200	-0.091893	0.187793	1.000000	-0.102157	-0.320114	-0.142832	0.115293	-0.111649	-0.229060	-0.155414	0.224604	-0.190824	0.315252	-0.159381	-0.220864
Q8	0.021471	0.107360	0.220369	0.051826	-0.002173	-0.102157	1.000000	0.427186	0.007712	0.062100	0.132023	0.140445	0.236392	-0.097767	-0.072330	-0.103531	-0.009775	0.211156
Q9	0.001116	0.243594	0.300727	0.133615	0.051743	-0.320114	0.427186	1.000000	0.122517	-0.018153	0.072504	0.107051	0.081738	-0.077491	0.127553	-0.102570	0.082830	0.096496
Q5_Partner	-0.060875	0.165788	0.034619	0.158118	0.048369	-0.142832	0.007712	0.122517	1.000000	-0.040323	0.141702	-0.054288	-0.064492	-0.005494	0.163062	-0.023172	0.159207	-0.037561
Q5_Friends	0.005929	-0.024624	0.099752	-0.001680	0.178767	0.115293	0.062100	-0.018153	-0.040323	1.000000	0.173496	0.016264	0.037394	0.062492	-0.096148	0.082531	-0.085508	0.000588
Q5_Siblings	0.045925	0.128528	0.127167	0.058572	-0.014079	-0.111649	0.132023	0.072504	0.141702	0.173496	1.000000	0.189603	0.095838	-0.052268	-0.050278	-0.067908	0.033451	0.097060
Q5_Co-worker	-0.031605	0.126227	0.203966	-0.058218	-0.125714	-0.229060	0.140445	0.107051	-0.054288	0.016264	0.189603	1.000000	0.286413	-0.117941	-0.168859	-0.243350	-0.113327	0.301263
Q6_Skyscraper	-0.007263	0.251940	0.288602	0.152946	-0.264792	-0.155414	0.236392	0.081738	-0.064492	0.037394	0.095838	0.286413	1.000000	-0.444683	-0.493692	-0.470483	-0.180539	0.601524
Q6_Sport	-0.012333	-0.277034	-0.153459	-0.150643	0.330094	0.224604	-0.097767	-0.077491	-0.005494	0.062492	-0.052268	-0.117941	-0.444683	1.000000	0.199375	0.376103	-0.151735	-0.403235
Q6_Art_and_Music	0.017180	0.049725	-0.029904	0.056070	0.231753	-0.190824	-0.072330	0.127553	0.163062	-0.096148	-0.050278	-0.168859	-0.493692	0.199375	1.000000	0.223406	0.301606	-0.364776
Q6_Carnival	-0.000061	-0.255610	-0.145800	-0.119064	0.490247	0.315252	-0.103531	-0.102570	-0.023172	0.082531	-0.067908	-0.243350	-0.470483	0.376103	0.223406	1.000000	-0.017433	-0.459600
Q6_Cuisine	-0.045108	0.174506	0.012567	0.192120	-0.038573	-0.159381	-0.009775	0.082830	0.159207	-0.085508	0.033451	-0.113327	-0.180539	-0.151735	0.301606	-0.017433	1.000000	-0.068078
Q6_Economics	-0.009231	0.306081	0.272768	0.145907	-0.253690	-0.220864	0.211156	0.096496	-0.037561	0.000588	0.097060	0.301263	0.601524	-0.403235	-0.364776	-0.459600	-0.068078	1.000000

Figure 1: Correlation Graph

Secondly, we employed box plots and correlation plots to understand how features relate to the target variable. For categorical features (Q1, Q2, Q3, Q4), we used cross tabulation to observe their distributions across different labels. Notably, each label displayed distinct peaks for certain rankings, suggesting meaningful correlations between these features and the target labels. This analysis confirmed that all categorical features contribute valuable information to the model.

```
Q1              1.0  2.0  3.0  4.0  5.0
Label
Dubai            5    5   41  162  152
New York City    2    4    5   35  313
Paris            1    5   11   72  274
Rio de Janeiro   7   24  142  136   53
```

Figure 2: Q1 cross tabulation

```
Q2              1.0  2.0  3.0  4.0  5.0
Label
Dubai           22   58  102  117   66
New York City    3   18   22   83  233
Paris           17   65  120   93   68
Rio de Janeiro  24  119  117   83   19
```

Figure 3: Q2 cross tabulation

```
Q3              1.0  2.0  3.0   4.0  5.0
Label
Dubai            14   25   50   124  152
New York City    18   58   96   111   76
Paris             8   12   47   118  178
Rio de Janeiro   16   86  138    85   37
```

Figure 4: Q3 cross tabulation

```
Q4              1.0  2.0  3.0   4.0  5.0
Label
Dubai            60  106   96    67   36
New York City    30   76   93    80   80
Paris            29   99  107    81   47
Rio de Janeiro    3    8   37    82  232
```

Figure 5: Q4 cross tabulation

For numerical features (Q7, Q8, Q9), we initially encountered difficulties in interpreting the box plots due to outliers. After removing outliers, we analyzed the plots for each label. Q7 exhibited significant variance across labels, indicating its potential as a strong indicator. However, Q8 showed considerable overlap between the box plots of "Dubai" and "New York City," suggesting that this feature might not provide significant discriminatory power and could be considered for pruning. In contrast, Q9 demonstrated clear separation between labels, indicating its potential usefulness in the model.
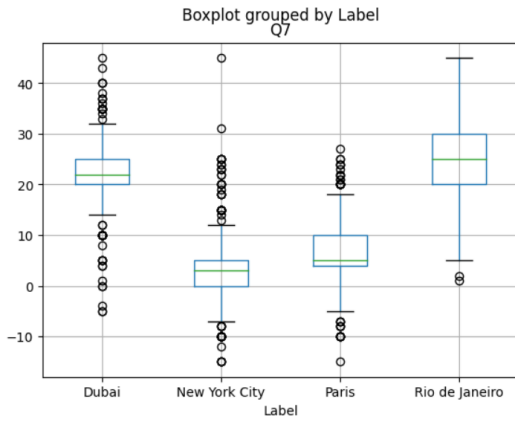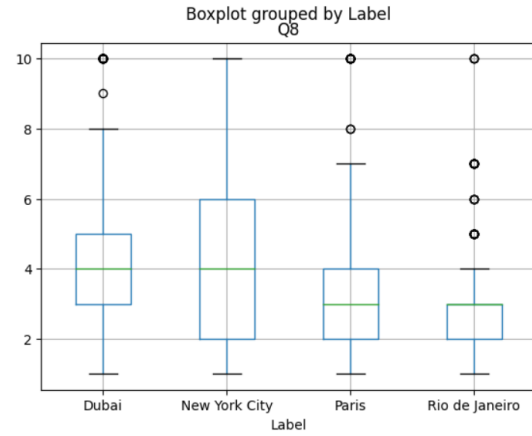

Figure 6: Q7 box plot
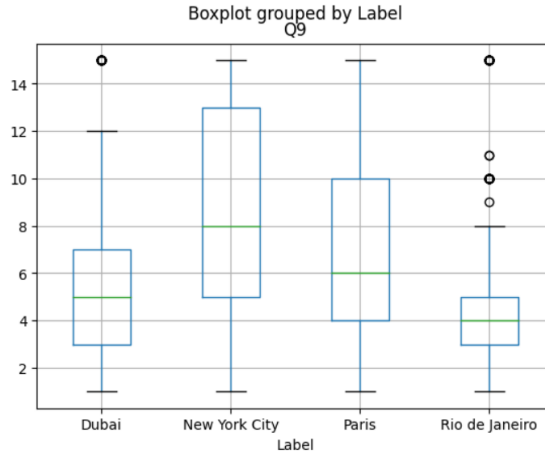

Figure 7: Q8 box plot

Figure 8: Q9 box plot

Since Q10 contained text data, we decided to represent it as a bag of words. The process of doing so resulted in a challenge during the generation of vocabulary. Each new word added a new feature, resulting in excessively large weight matrices and extended computation times. To address this issue, w e implemented two methods to reduce the vocabulary size: firstly, we removed stopwords such as "the", "there", "this",

etc. as these words often do not contribute significantly to sentence analysis and appear frequently, thereby disproportionately affecting the weights without adding value. We anticipated that by removing stopwords, the size of the matrices would decrease, and the analysis would become more efficient. Additionally, we eliminated words with a frequency of one or less. The hope is that these words are less importation to model performance since they show up so rarely. This decision will have reduced the overall model accuracy, as some informative vocabulary may have been pruned, but we deemed this trade-off as worth it due to the significant file size savings. In our test set, the model performance decreased by roughly 1%, but the file size of the weights decreased from over 10MB to below 4MB.
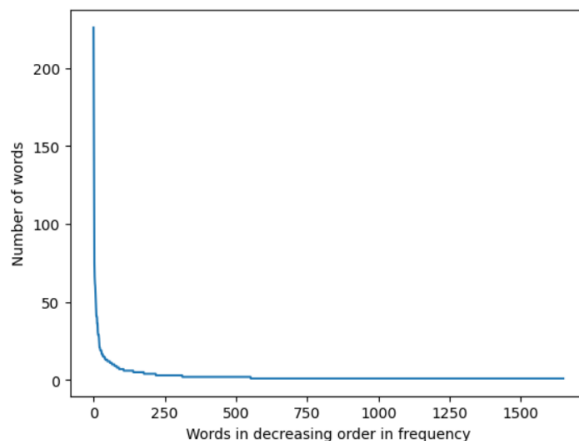

Figure 9: Vocabulary Distribution

Data Representation

In representing the data, we chose to convert categorical features (ie., Q1, Q2, Q3, Q4) into one-hot vectors to better perform operations on them. The conversion ensured that each category within a feature is represented by a binary value, making it easier for the model to process. Additionally, for questions 5 and 6, values were transformed into separate columns for each item to be ranked. For example, in Q6, columns were created for skyscrapers, sports, arts and music, carnival, cuisine, and economics, with each column representing the corresponding rank. Since Q7, Q8, and Q9 are continuous numerical features, they did not require conversion and could be used as is. Furthermore, to incorporate Q10 (which contains text) into our model, we converted the data into a bag of words representation to enable the model to process it.

Models Explored

k-Nearest Neighbours (kNN)

The first model we considered using for this challenge was one based on kNN due to the simplicity of the model. The concept of majority-vote classification would have been simple to implement and provided "good enough" predictions, but testing provided immediate issues with this kind of model.

We noticed that kNN was sensitive to noise, and the number of fields reduced accuracy - hence the simplicity of the model was also its Achilles' Heel. In particular, when testing with smaller k values, the accuracy remained satisfactory but would have increased overfitting. Yet when testing with larger k values, overfitting decreased but so did the accuracy of the predictions. This behaviour could be attributed to the large number of features present in the dataset, combined with the sparsity of the data itself.

As previously mentioned, kNN struggles to maintain accuracy when working with larger datasets. Since kNN calculates the distances between data points, large distances result in inefficient computation times and unnecessarily high consumption of storage space. Another problem that arises is the aforementioned large

number of fields. As the number of fields increases, datapoints clump together due to the distance between each point being reduced, leading to greater inaccuracy in predictions. This results in difficulty for the model to correctly classify each point.

Decision Tree Classifier

The Decision Tree Classifier was the second model we considered due to its robustness and multi-class classification properties. Given the amount of data provided, we felt that it was enough to give this model a try. The number of fields in the dataset plays to the Decision Tree Classifier's strengths, as it give it plenty of ways to split the data. Initial results of the Decision Tree Classifier proved to be fruitful: the prediction accuracy was fairly good at even lower max depths. However, increasing the max depth showed little to no improvements in the validation accuracy, and switching to various criterion proved to be more or less the same.

Additionally, it was difficult to find a good way to include the textual data, resulting in a significant loss in valuable data. These factors result in a model that is okay at generalizing each class, but not precise enough.

```
max_depth=4 Training Accuracy: 0.8333333333333334
max_depth=4 Validation Accuracy: 0.8316151202749141
max_depth=5 Training Accuracy: 0.8713298791018999
max_depth=5 Validation Accuracy: 0.8281786941580757
max_depth=6 Training Accuracy: 0.9196891191709845
max_depth=6 Validation Accuracy: 0.8316151202749141
max_depth=7 Training Accuracy: 0.957685664939551
max_depth=7 Validation Accuracy: 0.8213058419243986
max_depth=8 Training Accuracy: 0.9861830742659758
max_depth=8 Validation Accuracy: 0.8041237113402062
max_depth=9 Training Accuracy: 0.9965457685664939
max_depth=9 Validation Accuracy: 0.7972508591065293
```

Figure 10: Accuracy of Decision Tree Across Various Depths

Multi-Layer Perceptron (MLP)

Multi-Layer Perceptron was the third model we considered due to its scalability and generalization ability. Ultimately, it became our eventual final model choice. The model was trained on data in the form of one-hot vectors described above. In addition, we considered utilizing Naive Bayes to analyze the textual information present in Q10 and then combining it with the MLP model trained on the other features using an ensembling method. This approach would have reduced the weights of the MLP model by avoiding treating each word in Q10 as a feature. However, we ultimately decided against this idea in order to maintain the shared correlation between the other features and Q10. More information surrounding the implementation of the MLP model is provided later in the report (see Final Model Choice and Hyperparameters).

Implementation Considerations

k-NN

For the k-NN model, due to its simplicity, the hyperparameter that was optimized was the value of k, which determines the number of "nearest neighbours" used in classifying a datapoint. When selecting the value of k, the size of the dataset needed to be considered. Typically, a value of k around $\sqrt{n}$, where n is the number of datapoints, is preferred. Initially, values of k were tested to assess their impact on the overall accuracy. After trialing various k values, the optimal k value was approximately 17, an evaluation metric based on prediction accuracy. While the peak accuracy was at a k value of 5, choosing to use that value would likely lead to over fitting of the data due to the small number. On the other end of the graph, we can see the accuracy start decreasing significantly, where the model likely starts under fitting the data.
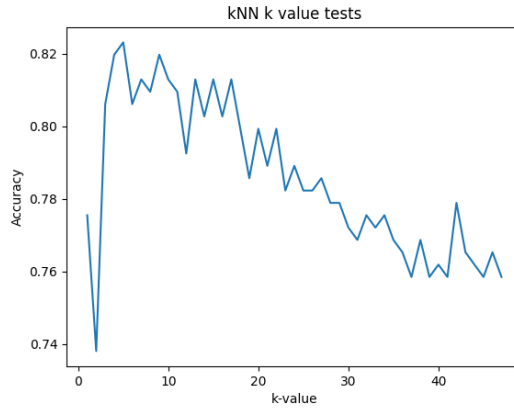
Figure 11: k-NN Model Accuracy Based on k-value

Decision Tree Classifier

The Decision Tree Classifier had two hyperparameters, those being the criterion and the max depth. Since there are 4 classes, a max depth of at least 2 is required for the there to be enough leaf nodes for the classification to work. Tuning these hyperparameters proved to have little to no effect on the final result, although around max depth of 4 to 6 seemed to work well. We tried to think of various ways to include the textual data in, such as a bag-of-words type approach, but it was difficult to see how this data would fit in the context of Decision Tree Classifiers due to the simple nature of the splitting in the Decision Tree Classifier and the complexity that the textual data insinuates.

MLP

The Multi-Layer Perceptron (MLP) model involved tuning several hyperparameters, specifically the hidden layer sizes, the number of hidden layers, the strength of the L2 regularizer, the learning rate, the activation function, and the method used for gradient descent. The number and size of hidden layers were particularly limited due to the large size of the input representation of Q10 as a bag of words. This size constraint was imposed to prevent the weights from becoming excessively large and exceeding the maximum file size. Consequently, the hyperparameter tuning process did not consider more than two hidden layers. The L2 regularizer was set on an interval of [0.001, 0.07].

The learning rates were either ['constant','adaptive', 'invscaling'].

The activation functions tested were ['tanh', 'relu', 'logistic'].

The method used for gradient descent was ['sgd', 'adam'].

Finally, the hidden layer size was tested on [(300,), (300, 300), (400,) (200, 200)].

The model was first fitted on a training set and with the baseline parameters of
(hidden layer shape: (300,), regularizer: 0.0001, invscaling, tanh, adam). Initial testing was performed on individual parameters to assess their impact on the output accuracy. Once the search space was narrowed down, the hyperparameter tuning was conducted on all parameters simultaneously, resulting in the highest accuracy with the following values:
(hidden layers:(100,100,100),max iter=3500,activation:tanh,random state:2,learning rate:adaptive)

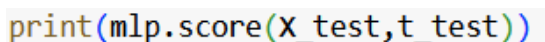# Final Model Choice and Hyperparameters

Upon completion of hyperparameter tuning, the accuracy of the three models were compared to determine the most accurate. To ensure consistent results among all three models, the same test set was used by all three, as well as the same random state values when using random sampling. This method provided fairness

and an equal chance for each of the models. In the end, the MLP model returned the greatest prediction accuracy, and therefore was deemed as our final model choice for implementation in pred.py. The model in pred.py is an MLP with 1 hidden layer and 1 output layer. The activation function used on the hidden layer is tanh, and the activation function used on the output layer is softmax. The model in pred.py also follows the general practices described above for data cleaning and pruning.

Predictions

The implemented model provided an accuracy rate of 89% when trialed on our test set. To ensure consistent and accurate testing, the data was split by label and then randomly selecting a testing set for each label equal to 20% of the total data. We constructed the vocabulary set for the bag of words based only on the training set to replicate working with an actual test set - this way, if any words appeared within the test set that were not present within the training set, we could determine how our model would react.

Since these conditions were created to mimic performance on an unseen test set, much like the one to be utilized by instructors during grading, we feel that our model will have similar accuracy to our testing results. We estimate around 88% accuracy for the reasons outlined above. Below is an image depicting the calling of tests and the resulting accuracy:

```
print(mlp.score(X_test,t_test))
```

0.88848920863330936

Figure 12: Calling Test

Figure 13: Resulting Accuracy upon Calling Test

Conclusion

In conclusion, this lab report detailed the thorough sorting and handling of the provided dataset, ensuring the data was clean and ready for analysis. Various methods were employed to address issues such as empty fields and outlier values, ensuring that the dataset was suitable for modeling. Additionally, the process of splitting the data into different sets using k-fold cross-validation helped to prevent bias and provided a robust evaluation of the model's performance.

During model exploration, the k-NN model exhibited simplicity but struggled to maintain accuracy due to the size of the dataset and its feature complexity, leading to issues with overfitting. The decision tree classifier provided a decent starting accuracy, however the accuracy did not improve significantly even with hyperparameter tuning. In addition, representing the data in Q10 was difficult with this model, resulting in excessive loss of valuable data. In contrast, the MLP was chosen as the final model due to its scalability and ability to capture complex relationships within the data. Despite initial challenges with the size of the input representation for Q10, careful andling and feature analysis ensured that the MLP model could effectively process the data and achieve a high accuracy.

The implementation considerations, including hyperparameter tuning and evaluation metrics, were crucial in selecting the final model. The MLP model, with optimized hyperparameters and careful evaluation, achieved an accuracy rate of 89% on the partitioned test set and is expected to perform similarly on unseen data due to the reasons provided throughout the report.