Group 17 – CS 4641 Group Project Final Report

Intro + Background

Pokémon is the classic Japanese role-playing game that pits trainers and their Pokémon against each other in battles. In this project, we aim to create a model that will take in 2 Pokémon, and information about them (such as their typing, stats, height, weight, moves, and abilities) and predict which of the two Pokemon would win against each other in a battle.

Problem Definition

We have found a dataset on Kaggle[1], which contains information on 50000 battles. Each battle contains information on two Pokémon, and the outcome of the battle. The dataset also contains information on every Pokémon, such as their base health, attack, defense, and types. We will train, using supervised learning, models that take in two Pokémon and solve the binary classification problem of predicting which Pokémon will win in battle. Accuracy and f1 score will be used to measure the performance of various models.

Method #1: Decision Tree

Our first attempt at exploring our problem was using decision trees. We wanted to start with decision trees for their simplicity and explainability. To create our decision tree model, first we preprocessed our data by giving each column of our data set a name (stats of a Pokémon such as attack, speed, defense, etc..) and dropping a redundant column that specified if they were the first Pokémon or second Pokémon in a battle.

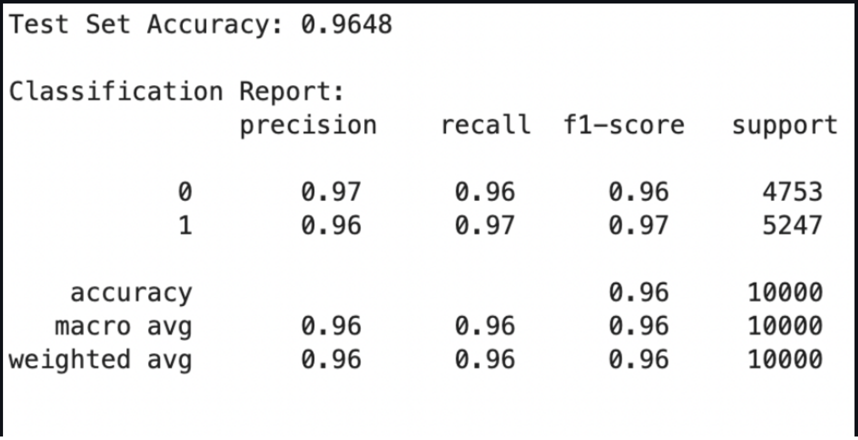
We used a one-hot encoder to represent the Pokémon types (1 encoder for each Pokémon).

For our feature engineering, we used the given stats and type of each Pokémon (8 for each. 2 types + 5 stats + legendary or regular Pokémon status). We also calculated the difference between stats of both Pokémon to use as features, creating an additional 7 features.

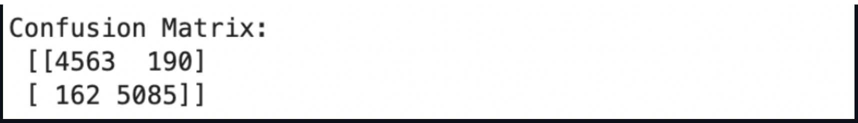
Our model was then trained using scikit-learn's decision tree classifier on 80% of the data as the training data. Metrics were calculated using the remaining 20% of the data, the test data.

Decision Tree Results

Our Decision Tree approach had a very high cross-validation accuracy at around 96%. The 96% f1 score is also very high. (Fig. 1.)



```
Test Set Accuracy: 0.9648

Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.96      0.96      4753
           1       0.96      0.97      0.97      5247

    accuracy                           0.96     10000
   macro avg       0.96      0.96      0.96     10000
weighted avg       0.96      0.96      0.96     10000
```



```
Confusion Matrix:
 [[4563  190]
 [ 162 5085]]
```

Fig.1. Accuracy and metrics of Decision Tree approach.

We found that speed is overwhelmingly the most important feature at about 80% (Fig 2.), which makes sense because if you go first, you get an extra hit before your opponent. Every other feature is insignificant, with attack_diff being the only one poking out a little at about 3%. We also found that assigning importance to other attributes like Legendary and attack_diff did not change much, as the decision tree was stubborn to it with speed_diff being overwhelmingly the best indicator. After dividing the value of speed_diff by 3, it stayed the same again. I even tried combining speed and attack since that increases chances of a one hit knockout, but that had about 91% accuracy, not improving it.

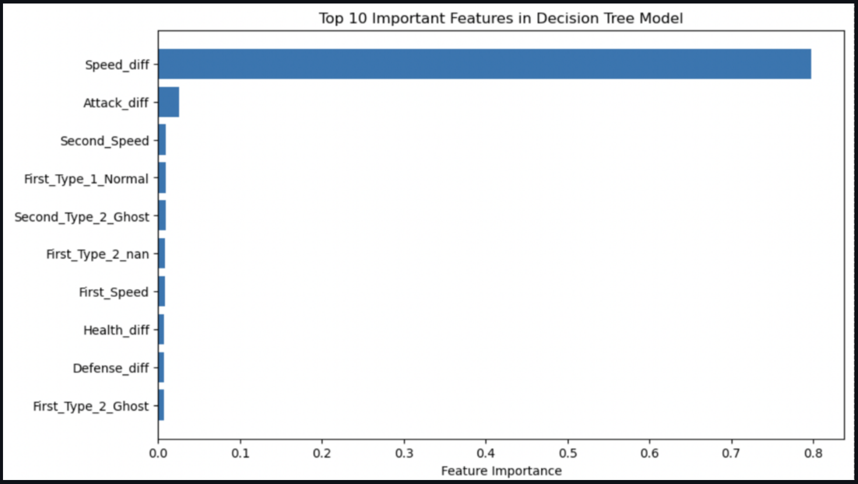


Fig 2. Top 10 important features in decision tree model.

Without speed, attack and special attack are the most important, with the rest of the attributes being essentially tied (Fig. 3.). This is not very telling data since we dropped speed_diff and accuracy dropped about 25% as a result, but it gives an idea of what is being hidden by the overwhelming dominance of speed. In conclusion, even after exploring changes to weights and labels, speed is by far the most important feature in determining who would win with decision trees.

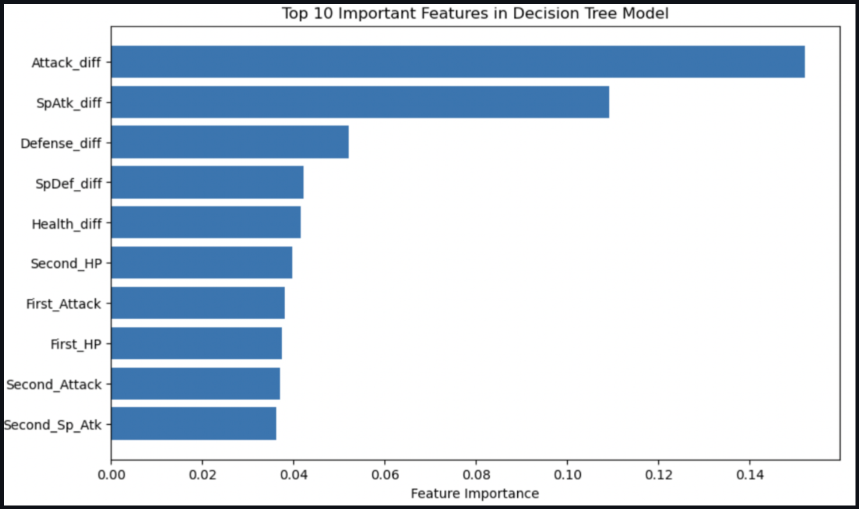


Fig. 3. Top 10 most important features without speed.

Method #2: Random Forests

Random Forests build upon decision trees by using a multitude of decision trees in an ensemble through bagging.

To build a random forest, we train one decision tree using only a subset of the data (sampled with replacement) using a subset of the features (sampled without replacement).

This is done multiple times to build a collection of multiple decision trees. The final classification is decided using a majority vote of the decision trees.

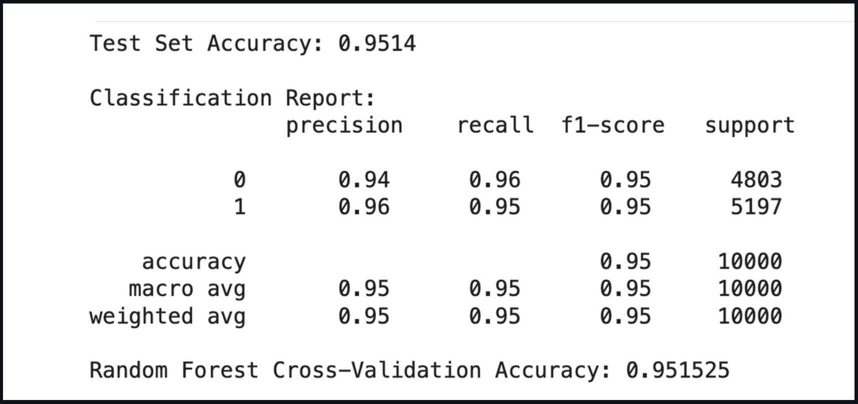
We preprocessed the data using the same way as the decision trees, and did feature engineering similarly.

The random forest model was then trained using scikit-learn's random forest classifier on 80% of the data as the training data. Metrics were calculated using the remaining 20% of the data, the test data.

As Figure 4 below shows, the random forest approach has a slightly lower performance than the decision tree approach, having a test set accuracy of 95.14% compared to the decision tree’s 96.48%.

This could be due to random forests generally having a slightly higher bias, in return for much lower chance of overfitting. However, since the test set accuracy and the F1-score of the decision trees were already so high, there was likely very little overfitting occurring.

Hence, the random forest approach could not provide a higher accuracy or performance compared to the decision tree, and the added bias contributed to the slightly lower accuracy.



```
Test Set Accuracy: 0.9514

Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.96      0.95      4803
           1       0.96      0.95      0.95      5197

    accuracy                           0.95     10000
   macro avg       0.95      0.95      0.95     10000
weighted avg       0.95      0.95      0.95     10000

Random Forest Cross-Validation Accuracy: 0.951525
```

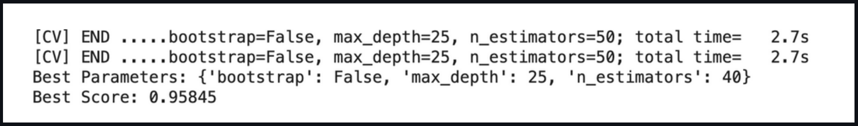
Fig 4. Accuracy and metrics of the random forest approach.

To try to improve the performance of the random forest, we proceeded with hyperparameter tuning, which attempts to find the best set of hyperparameters for the random forest.

Hyperparameter tuning was done with sklearn’s GridSearchCV function, which takes in a grid of candidate hyperparameters, then systematically fits a random forest on each possible combination of hyperparameters.

The performance of each random forest was measured in each trial using cross-validation, then the best score along with the associated set of hyperparameters are reported.

As shown in figure 6, the best accuracy was 95.85%, a modest improvement from the 95.14% of the previous set.



```
[CV] END .....bootstrap=False, max_depth=25, n_estimators=50; total time=   2.7s
[CV] END .....bootstrap=False, max_depth=25, n_estimators=50; total time=   2.7s
Best Parameters: {'bootstrap': False, 'max_depth': 25, 'n_estimators': 40}
Best Score: 0.95845
```

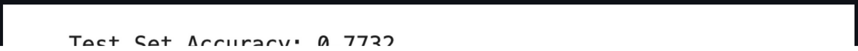
Fig 5. Score obtained using hyperparameter search. .

To further push the performance of the model, we also included additional features which could potentially provide additional information to aid the model in classification.

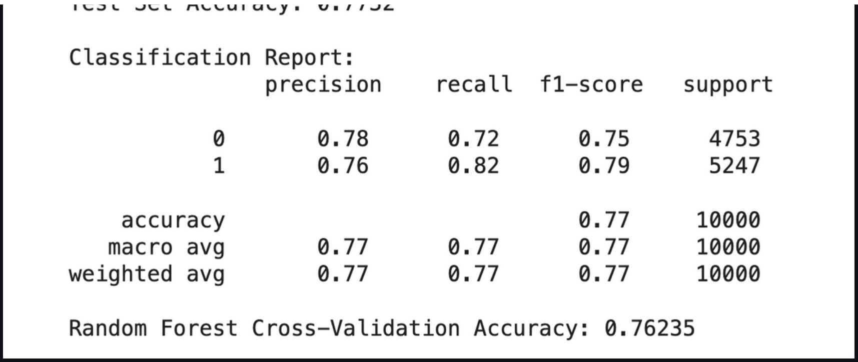
Specifically, we included the type of the Pokémon, which was one-hot-encoded, as well as the height and weight of the Pokémon.

Figure 6 below shows the results, which indicates a dramatic reduction in the accuracy and F1-score of the model from 95% to around 77%. This indicates that the additional features are likely uninformative and contributed mostly noise to the model.

This, in turn, resulted in the model likely overfitting to the noise features, resulting in a much degraded performance.



```
Test Set Accuracy: 0.7732
```



```
Test Set Accuracy: 0.7752

Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.72      0.75      4753
           1       0.76      0.82      0.79      5247

    accuracy                           0.77     10000
   macro avg       0.77      0.77      0.77     10000
weighted avg       0.77      0.77      0.77     10000

Random Forest Cross-Validation Accuracy: 0.76235
```

Fig. 6. Accuracy and metrics of the random forest approach with additional features.

Method #3: Logistic Regression

Logistic regression is an easy-to-interpret and effective model that is often used to classify values.

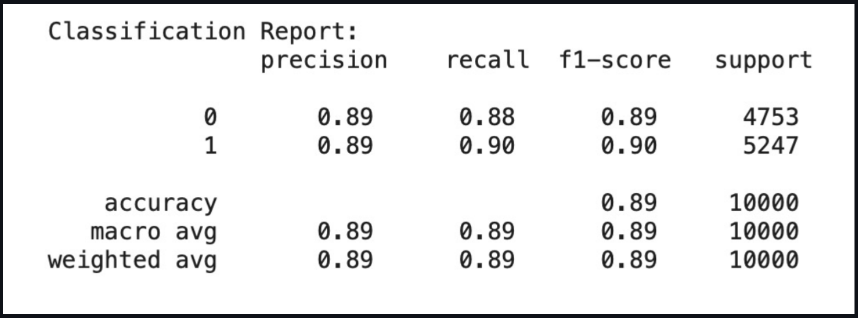
We wanted to include this method because of its efficiency and ability to avoid overfitting the training data. We used logistic regression to do a binary classification of wins/loss predictions on the Pokémon battle data. Each of the two battling Pokémon's types were included as features into the logistic regression model.

The sigmoid function was used to classify the battles into predicted wins or losses.

First, basic data cleaning was performed, and then one-hot coding was done to include categorical features and Pokémon types in the model.

The given Pokémon to predict from were type fitted and scaled to be able to be modeled off.

We utilized sklearn for the model in lieu of creating our own. We split the data into 80% for the training data and 20% for the test data.

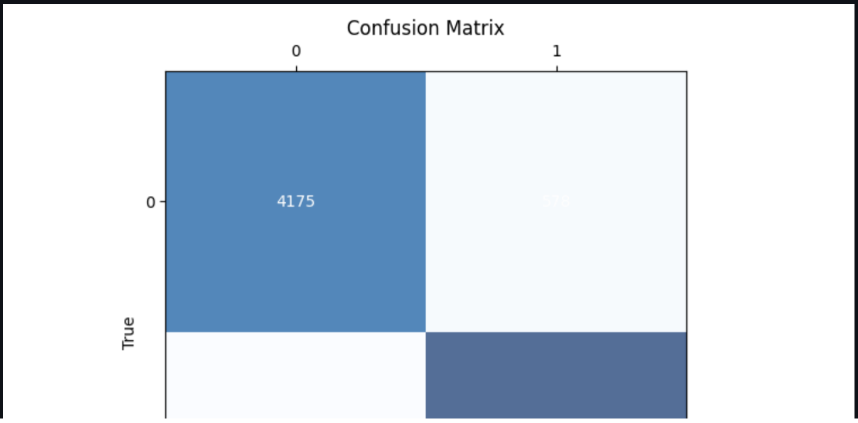


```
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.88      0.89      4753
           1       0.89      0.90      0.90      5247

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```

Fig.7. Accuracy and metrics of Logistic Regression approach.

As Figure 7 shows above, the logistic regression model had an accuracy of 89 percent. It was slightly more accurate when predicting wins than losses, but not by a significant amount.

The Confusion Matrix shown in Figure 8 also shows high accuracy, with just under 9000 out of the 10000 data points in the training set classified correctly as lost or won.



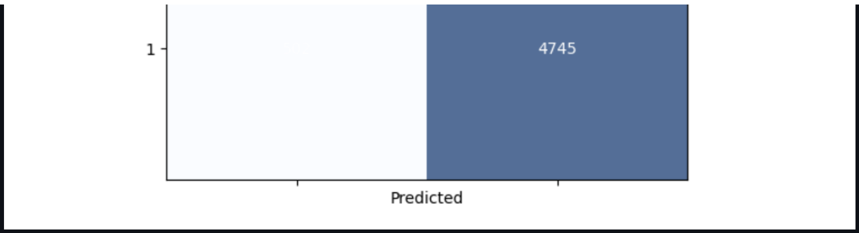


Fig. 8. Confusion matrix from the logistic regression model.

Discussion

As the results above show, using a decision tree has given us 96% cross-validation accuracy. Additionally, our F1 score for both outcomes on which Pokémon wins is around 96%, which means that our model has both high precision and recall.

We can also see that speed difference is the most important feature that decides the outcome of a battle, followed by attack and special attack.

We also tried using random forests and logistic regression. Random forests gave a performance (95%) on par with the decision trees or slightly worse, even with hyperparameter tuning, while logistic regression resulted in a worse score overall (90%).

It is likely that random forests were unable to confer many gains in terms of reducing overfitting, while slightly increasing bias, while logistic regression was underfitting the data and hence both methods resulted in worse performance than decision trees.

Conclusion

Decision trees were the best method, giving the best accuracy and F1 score, compared to random forests and logistic regression.

For next steps, we can consider engineering more features, such as using the Pokémon's types, and constructing a new feature based on whether Pokémon 1 is weak to Pokemon's 2 type or vice versa.

Additionally, if speed really is such a dominating feature, we could investigate heavily simplifying prediction models for this goal.

Youtube Link to Video

https://youtu.be/0oqmzWOfH0Y

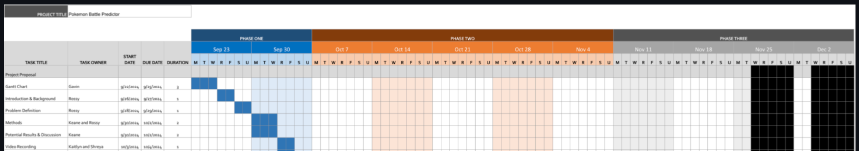
References

[1] "Pokemon - Weedle's Cave." Accessed: Oct. 02, 2024. [Online]. Available: https://www.kaggle.com/datasets/terminus7/pokemon-challenge

[2] A. Ng, "Preventing 'Overfitting' of Cross-Validation Data," Proceedings of the Fourteenth International Conference on Machine Learning, Jan. 1998.

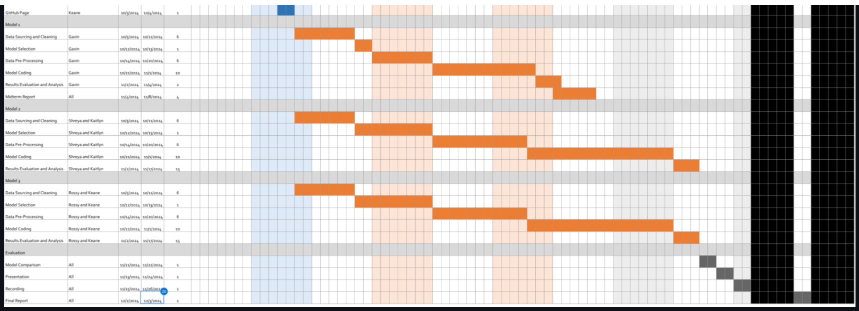
[3] R. Banik, "The Complete Pokemon Dataset," Kaggle https://www.kaggle.com/datasets/rounakbanik/pokemon (accessed Oct. 4, 2024).

Gantt Chart



PROJECT TITLE Pokemon Battle Predictor

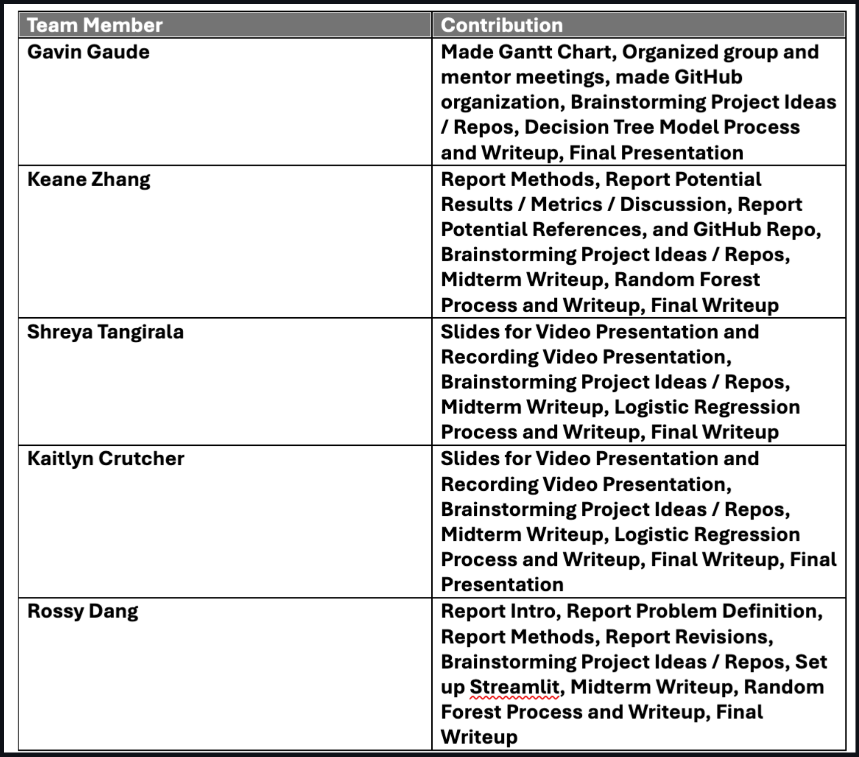
TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION
Project Proposal				
Gantt Chart	Gavin	9/22/2024	9/25/2024	3
Introduction & Background	Rossy	9/26/2024	9/27/2024	1
Problem Definition	Rossy	9/28/2024	9/29/2024	1
Methods	Keane and Rossy	9/30/2024	10/2/2024	2
Potential Results & Discussion	Keane	9/30/2024	10/2/2024	2
Video Recording	Kaitlyn and Shreya	10/3/2024	10/4/2024	1



GitHub Page	Keane	10/3/2024	10/4/2024	1
Model 1				
Data Sourcing and Cleaning	Gavin	10/5/2024	10/11/2024	6
Model Selection	Gavin	10/12/2024	10/13/2024	1
Data Pre-Processing	Gavin	10/14/2024	10/20/2024	6
Model Coding	Gavin	10/21/2024	11/1/2024	10
Results Evaluation and Analysis	Gavin	11/2/2024	11/4/2024	2
Midterm Report	All	11/4/2024	11/8/2024	4
Model 2				
Data Sourcing and Cleaning	Shreya and Kaitlyn	10/5/2024	10/11/2024	6
Model Selection	Shreya and Kaitlyn	10/12/2024	10/13/2024	1
Data Pre-Processing	Shreya and Kaitlyn	10/14/2024	10/20/2024	6
Model Coding	Shreya and Kaitlyn	10/21/2024	11/1/2024	10
Results Evaluation and Analysis	Shreya and Kaitlyn	11/2/2024	11/17/2024	15
Model 3				
Data Sourcing and Cleaning	Rossy and Keane	10/5/2024	10/11/2024	6
Model Selection	Rossy and Keane	10/12/2024	10/13/2024	1
Data Pre-Processing	Rossy and Keane	10/14/2024	10/20/2024	6
Model Coding	Rossy and Keane	10/21/2024	11/1/2024	10
Results Evaluation and Analysis	Rossy and Keane	11/2/2024	11/17/2024	15
Evaluation				
Model Comparison	All	11/21/2024	11/22/2024	1
Presentation	All	11/23/2024	11/24/2024	1
Recording	All	11/25/2024	11/26/202	1
Final Report	All	12/2/2024	12/3/2024	1

Gantt Chart

Contribution Table



Team Member	Contribution
Gavin Gaude	Made Gantt Chart, Organized group and mentor meetings, made GitHub organization, Brainstorming Project Ideas / Repos, Decision Tree Model Process and Writeup, Final Presentation
Keane Zhang	Report Methods, Report Potential Results / Metrics / Discussion, Report Potential References, and GitHub Repo, Brainstorming Project Ideas / Repos, Midterm Writeup, Random Forest Process and Writeup, Final Writeup
Shreya Tangirala	Slides for Video Presentation and Recording Video Presentation, Brainstorming Project Ideas / Repos, Midterm Writeup, Logistic Regression Process and Writeup, Final Writeup
Kaitlyn Crutcher	Slides for Video Presentation and Recording Video Presentation, Brainstorming Project Ideas / Repos, Midterm Writeup, Logistic Regression Process and Writeup, Final Writeup, Final Presentation
Rossy Dang	Report Intro, Report Problem Definition, Report Methods, Report Revisions, Brainstorming Project Ideas / Repos, Set up Streamlit, Midterm Writeup, Random Forest Process and Writeup, Final Writeup

Contribution Table