"Exploring Online Chess Games Using a Data Science Based Approach"

Yadvender Singh
50466664
yadvende@buffalo.edu
Computer Science and Engineering
CSE 587
University at Buffalo
Buffalo, New York, USA

Shawna Saha
50468278
shawnasa@buffalo.edu
Computer Science and Engineering
CSE 587
University at Buffalo
Buffalo, New York, USA

I. Introduction

Since time immemorial Chess has been a prevalent game that has captivated the minds of a large audience. The highly esoteric nature of this game, along with its simple rules but overly complex and intricate plays make it an ideal sport for information extraction via data analysis. Data Science can be applied to chess game analysis to build player profiles, build fraud detection system in online chess games, recommender system for chess training and practice etc.

II. Problem Statement

Though the game of chess has been played for millennia, there is a lack of comprehensive understanding of the dynamics of the game. Determining the optimum set of moves to achieve victory is a significant struggle since there is no one right strategy to play the game. The game's arcane nature has made it a challenge even for the brightest brains.

With the development of technology acquiring data on chess games have become relatively feasible. By analyzing the enormous databases of chess games, data science approaches can help in spotting patterns and trends, create models that may forecast upcoming moves and results.

Thus, this project aims to employ data science techniques in the analysis of a large dataset of online chess games to gain insights into the patterns, trends and strategies used by different chess players at different levels of the game.

The primary questions we tried to answer through exploratory data analysis were:

1. What are the most common openings used by successful chess players and how does it affect their game?
2. Is it possible to predict the outcome of a game based on the opening move chosen by a player?
3. What sequence of moves have the largest probability to win?
4. How does a player's rating affect the choice of opening move and the outcome of the game?
5. How do players of similar ratings perform when played against each other?
6. Does a white player have an advantage in how the game unfolds due to the rule of the game that states – "White moves first"?

In Phase 2, by employing several machine learning algorithms, we were trying to make predictions about white being emerging as victorious in a game of chess.

Answering these questions will help in attaining the following objectives:

1. By analyzing the data to better understand a player's performance such as – number of wins, number of losses, preferred first move etc. can help in coaching and suggest improvements to the game.
2. Insights into the most common opening strategies, number of turns, sequence of moves, the most successful tactics etc. can help in creating better training materials.

In Phase 3, we developed a web application "Chess Analysis" by using Python and Streamlit. The potential user of this web application are chess coaches, chess players, online interactive chess game designers etc.

III. Data Sources

The dataset used for the implementation of the project are detailed below:
Name of the dataset: Chess Game Dataset (Lichess)
Source: www.kaggle.com
Link: https://www.kaggle.com/datasets/datasnaek/chess?datasetId=2321&searchQuery=eda

Description: This is a dataset containing data collected from 20,000 online chess games from the site Litchess.org. The dataset has 16 features in total. They are as detailed below:

1. ID: Game id.
2. RATED: If a player is rated or not.
3. CREATED AT: Start time of the game.
4. LAST MOVE AT: End time of the game.
5. TURNS: Number of turns in the game.
6. VICTORY STATUS: The outcome of the game.
7. WINNER: Which player won.
8. INCREMENT CODE: The amount of time added to the clock after each move made by a player.
9. WHITE ID: Id of the white player.
10. WHITE RATING: Rating of the white player.
11. BLACK ID: Id of the black player.
12. BLACK RATING: Rating of the white player.
13. MOVES: Set of moves made by the players in chess notation.
14. OPENING ECO: Standardized codes for the chess opening names.
15. OPENING NAME: The chess opening used by the player.
16. OPENING PLY: Number of moves in the opening phase.

IV. ALGORITHMS

The machine learning algorithms we used for the Chess Analysis Web Application are as follows:

Algorithms discussed in class:

1. **Naïve Bayes:**
   Naïve Bayes model is used with Bagging Classifier ensemble model with Gaussian Naïve Bayes as the base estimator. Number of estimators in the ensemble is set to 10. The Bagging Classifier aggregates the predictions from several Gaussian Naive Bayes model instances, improving predictive performance of the model.

2. **K-Nearest Neighbor:**
   K -Nearest Neighbor algorithm is used with 5, 15 and 20 neighbors with all neighbors having uniform weight. For hyperparameter tuning, GridSearchCV is used with 5-fold cross-validation strategy.

3. **Logistic Regression:**
   Logistic Regression model is used with no hyperparameter tuning.

Algorithms outside of class discussion:

4. **Decision Tree:**
   The hyperparameters used for Decision Tree model are ‘gini’ is used for Gini Impurity and ‘entropy’ is used for information gain with maximum depths - 4, 5, 6, 7, 8, 9, 10, 11, 12, 15, 20, 30, 40, 50, 70, 90, 120, 150. GridSearchCV is used for hyperparameter tuning with a 10-fold cross-validation strategy.

5. **Random Forest:**
   Random Forest model is used with no tuning.

6. **Neural Network - Multi-Layer Perceptron:**
   Limited-memory Broyden-Fletcher-Goldfarb-Shanno is used for optimization. The regularization parameter for weight decay is chosen as 0.00001. There are 2 hidden layers with 5 neurons each. Maximum number of epochs is used as 2000.

V. DATA CLEANING

The detailed process of cleaning the dataset is described as below. All these operations are performed in the backend of the application.

1. **Column Names Formatting**: Total number of rows in the original dataset is 20,058. At first, the column names were formatted for better visual representation.

```
Formatted column names with associated datatype are as folows :
ID                 object
RATED                bool
CREATED AT        float64
LAST MOVE AT      float64
TURNS               int64
VICTORY STATUS     object
WINNER             object
INCREMENT CODE     object
WHITE ID           object
WHITE RATING        int64
BLACK ID           object
BLACK RATING        int64
MOVES              object
OPENING ECO        object
OPENING NAME       object
OPENING PLY         int64
dtype: object
```

2. **Changing column datatype:** "CREATED AT" and "LAST MOVE AT" features were converted from object to datetime format.

```
ID                         object
RATED                        bool
CREATED AT        datetime64[ns]
LAST MOVE AT      datetime64[ns]
TURNS                      int64
VICTORY STATUS            object
WINNER                    object
INCREMENT CODE            object
WHITE ID                  object
WHITE RATING               int64
BLACK ID                  object
BLACK RATING               int64
MOVES                     object
OPENING ECO               object
OPENING NAME              object
OPENING PLY                int64
dtype: object
```

3. **Adding a column:** "MATCH DURATION" feature was added to store the duration of each Chess match.

   MATCH DURATION = LAST MOVE AT - CREATED AT

```
ID                          object
RATED                         bool
CREATED AT         datetime64[ns]
LAST MOVE AT       datetime64[ns]
TURNS                       int64
VICTORY STATUS             object
WINNER                     object
INCREMENT CODE             object
WHITE ID                   object
WHITE RATING                int64
BLACK ID                   object
BLACK RATING                int64
MOVES                      object
OPENING ECO                object
OPENING NAME               object
OPENING PLY                 int64
MATCH DURATION            float64
dtype: object
```

4. **Selecting features:** The features of interest were extracted from the dataframe. The list of features selected are:

```
RATED                bool
TURNS               int64
VICTORY STATUS     object
WINNER             object
INCREMENT CODE     object
WHITE ID           object
WHITE RATING        int64
BLACK ID           object
BLACK RATING        int64
MOVES              object
OPENING NAME       object
OPENING PLY         int64
MATCH DURATION    float64
dtype: object
```

5. **Dropped 'seconds' information from "INCREMENT CODE":** The column increment code contained data of form "10+2" where the part after '+' sign represented seconds. For time increment we are only interested in the minutes hence formatted the column by splitting the data and keeping only the minutes information. Then datatype of the column was changed from object to Integer.

```
print(game_data['INCREMENT CODE'].head(5))
```

```
0    15
1     5
2     5
3    20
4    30
Name: INCREMENT CODE, dtype: int64
```

6. **Checking for null values:** There were no null values in the code, hence null removal operation was not required.
7. **Removing duplicates:** There was duplicate data in the dataset, so the duplicate columns were dropped. Data count after removal of duplicates was 19,629 records.
8. **Dropping games shorter than 2 moves:** The shortest possible checkmate in Chess is "Fool's mate" where the black player wins after two moves. Any number of turns less than 2 is erroneous and insignificant. Hence data containing turns less than 2 were dropped. Data count after this operation was 19,345.
9. **Converting column to list:** Each entry in the MOVES column was converted to list for easy access in future.
10. **Extracting Opening Categories:** OPENING NAMES contained the main category of opening names along with the subcategories mentioned. Hence, the main category of opening names was extracted and saved in a new column "OPENING CATEGORY" for better analysis.

```
Sicilian Defense       2508
French Defense         1269
Queen's Pawn Game      1017
Italian Game            953
King's Pawn Game        869
                        ...
Alekhine Defense #3       1
Doery Defense             1
Petrov's Defense #5       1
Global Opening            1
Pterodactyl Defense       1
Name: OPENING CATEGORY, Length: 180, dtype: int64
```

11. **Adding "MEAN RATING" column:** MEAN RATING column was added to the dataset.

MEAN RATING = (WHITE RATING + BLACK RATING) / 2

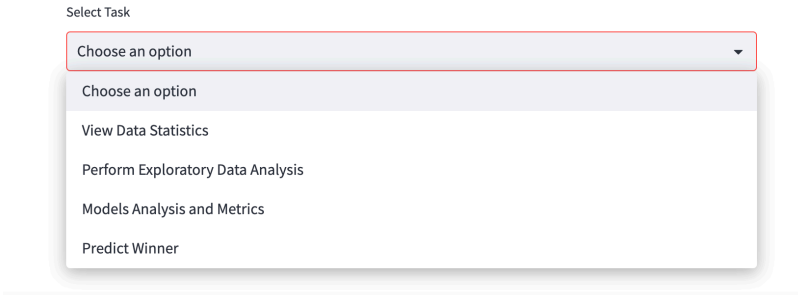
12. **Diving the game into levels:** The game was divided into 4 levels namely- Beginner, Intermediate, Advanced and Expert based on the rating distribution. This information was stored in a new column "GAME LEVEL".

```
Beginner : 800 to 1225
Intermediate : 1225 to 1650
Advanced : 1650 to 2075
Expert : 2075 to 2500
```

13. **Removing columns having 0 game duration:** On exploratory data analysis it was found that 25% of the MATCH DURATION data had 0 values which was erroneous. Hence only those columns were retained where MATCH DURATION was greater than 0. Data count after this operation was 10,885 records.
14. **Segregating games into variants:** The data was segregated into four variants based on the increment time: Bullet, Blitz, Rapid and Classical and the information was stored in a new column "GAME VARIANT".
15. **Removing least used openings:** On exploratory data analysis it was observed that 25% of the openings were used only once. Hence openings that were used less than 20 times were removed. The count of data after this operation was 7,672 records.

VI.DATA PRE-PROCESSING FOR MODEL FITTING

The following pr-processing steps were followed to process the data further in order to fit the machine learning models. All pre-processing steps were done in the backend once the user chooses 'Exploratory Data Analysis' and 'Model Analysis and Metrics' operations to perform from the user interface.



a. **Dropping "draw" outcomes:** Since we are only considered about the chess piece color of the winner of the game, we are dropping outcomes that result to a draw.

b. **Label Encoding:** Label Encoder was used to transform categorical features, namely – WINNER, OPENING CATEGORY, GAME LEVEL, GAME VARIANT

The outcome variable WINNER was encoded as:

0 = Black

1 = White

c. **Choosing Features for Prediction:** The final list of chosen features used as predictor variables are:
`RATED, TURNS, INCREMENT CODE, WHITE RATING, BLACK RATING, OPENING CATEGORY, MEAN RATING, GAME LEVEL, GAME VARIANT.`

d. **Splitting data into training data and testing data:** The dataset can be split using the slider window in the User Interface and the user can modify it as per their requirements. Modification of the data splitting slider will influence the predictions by the model.

e. **Normalizing and Scaling Data:** The data was normalized and scaled using methods from sklearn library in the backend.

VII. LIST OF OPERATIONS AVAILABLE

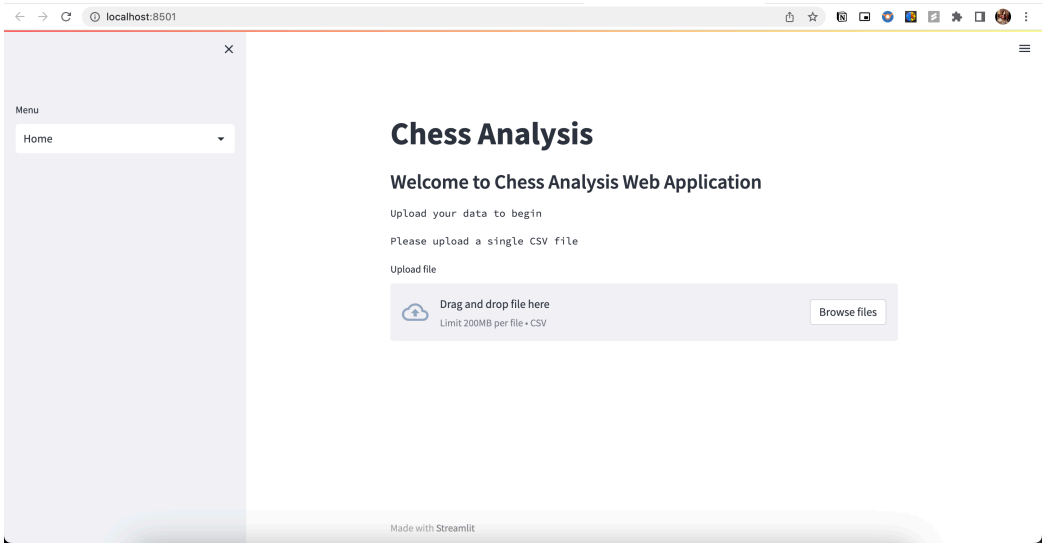
The list of operations a user can perform in the web application are as follows:

1. Upload a dataset.
2. View the statistical details of the dataset.
3. Perform exploratory data analysis.
4. Analyze the models based on their evaluation metrics.
5. Predict winner by entering custom data.

VIII. USER INTERFACE

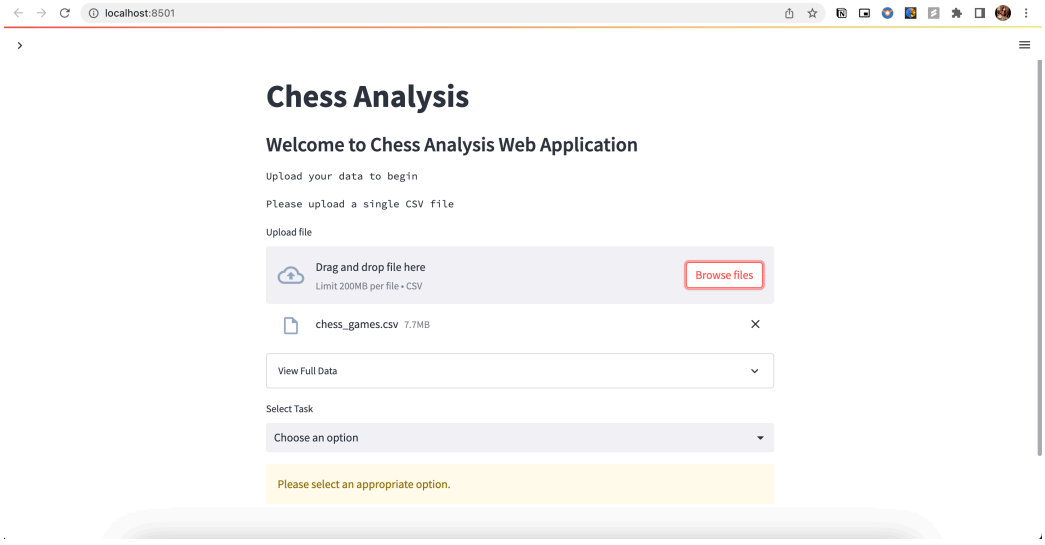
Home:

This is the homepage of the application.



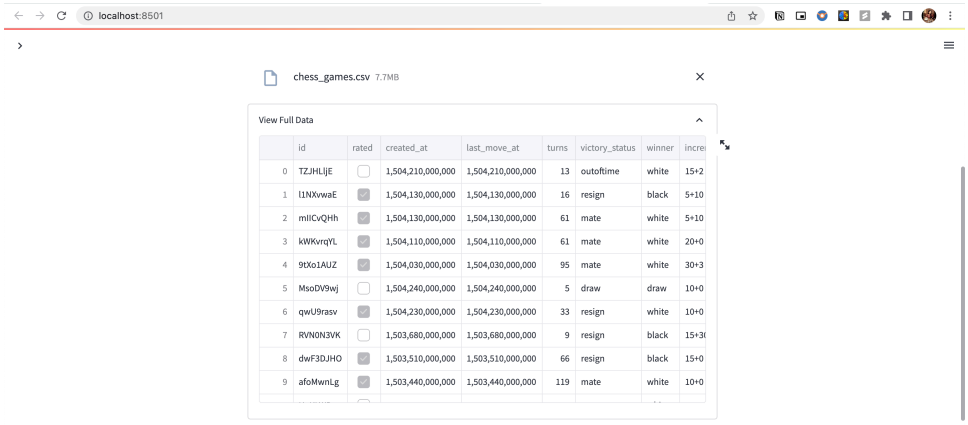
Upload Data:

.csv files of data can be uploaded in the user interface. The data uploaded must have the features: ID, RATED, CREATED AT, LAST MOVE AT, TURNS, VICTORY STATUS, WINNER, INCREMENT CODE, WHITE ID, WHITE RATING, BLACK ID, BLACK RATING, MOVES, OPENING ECO, OPENING NAME, OPENING PLY.



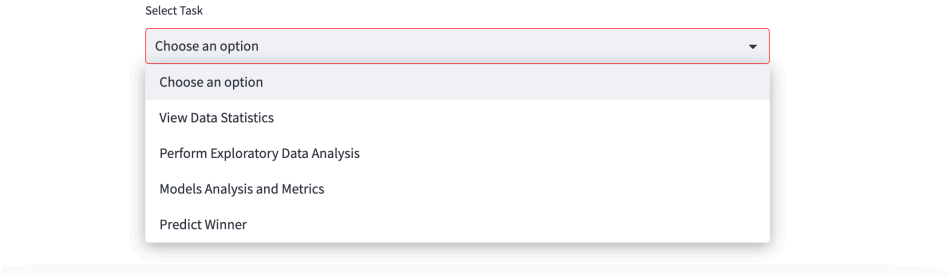
View Full Data:

The dataset can be viewed with the help of this option.



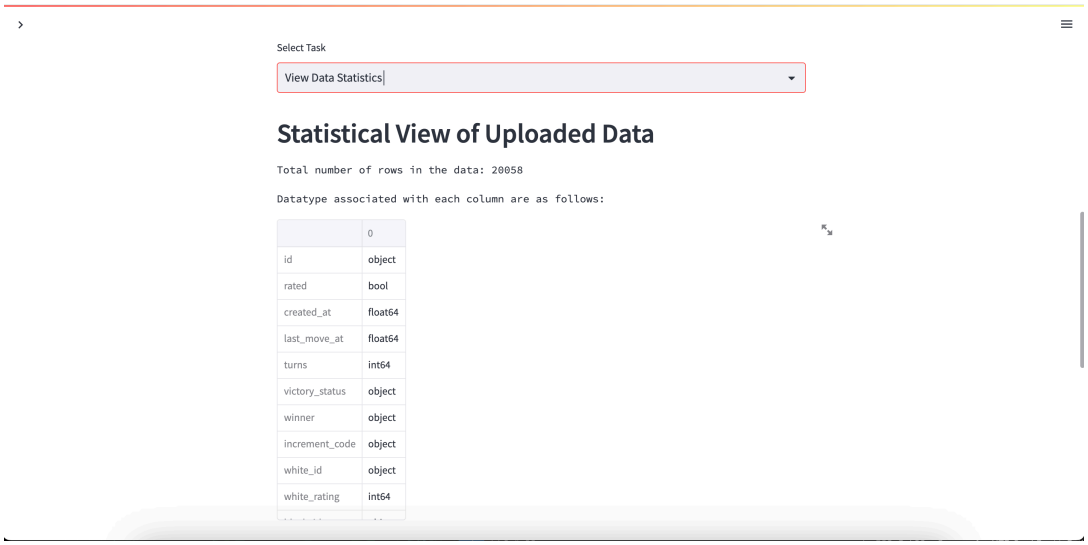
Select Operation to Perform:

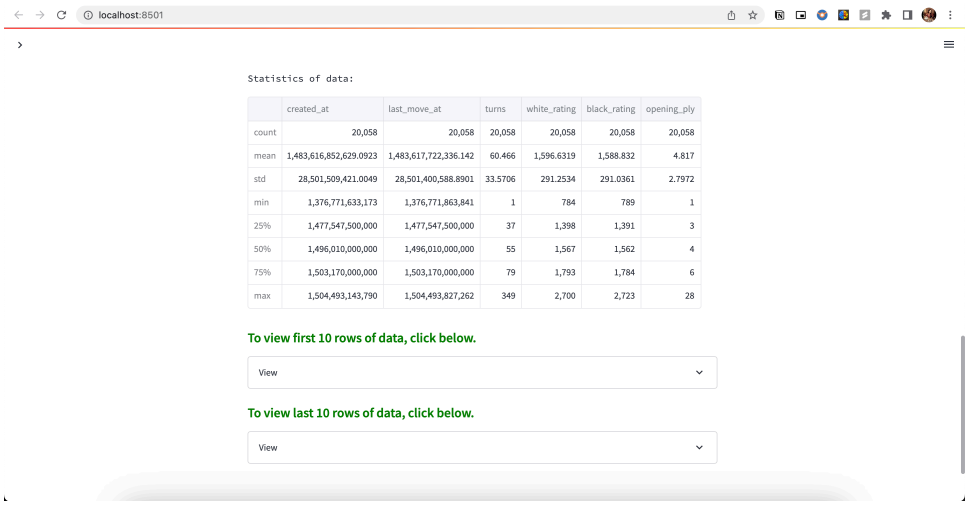
The user can select the operation they want to perform on the data.

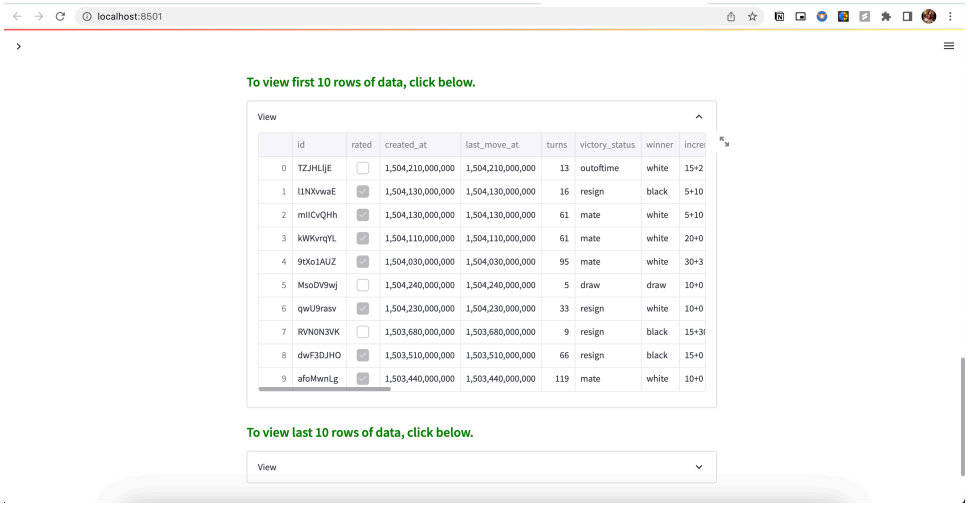


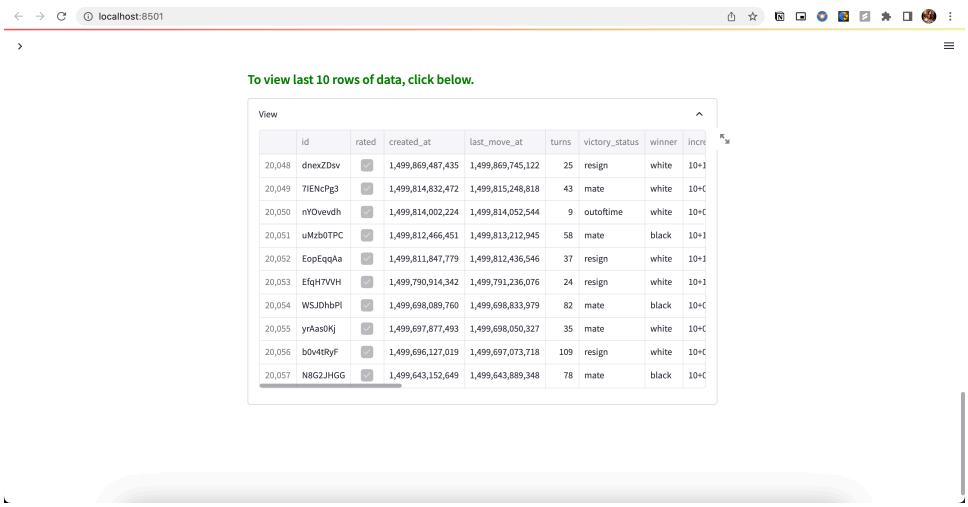
View Data Statistics:

The user can view the details of the dataset from this screen such as: the datatypes associated with each feature, the number of rows present in the data, the distribution of the data, how the first and the last 10 rows of the data look. This was the user can have an idea of the data they have uploaded.



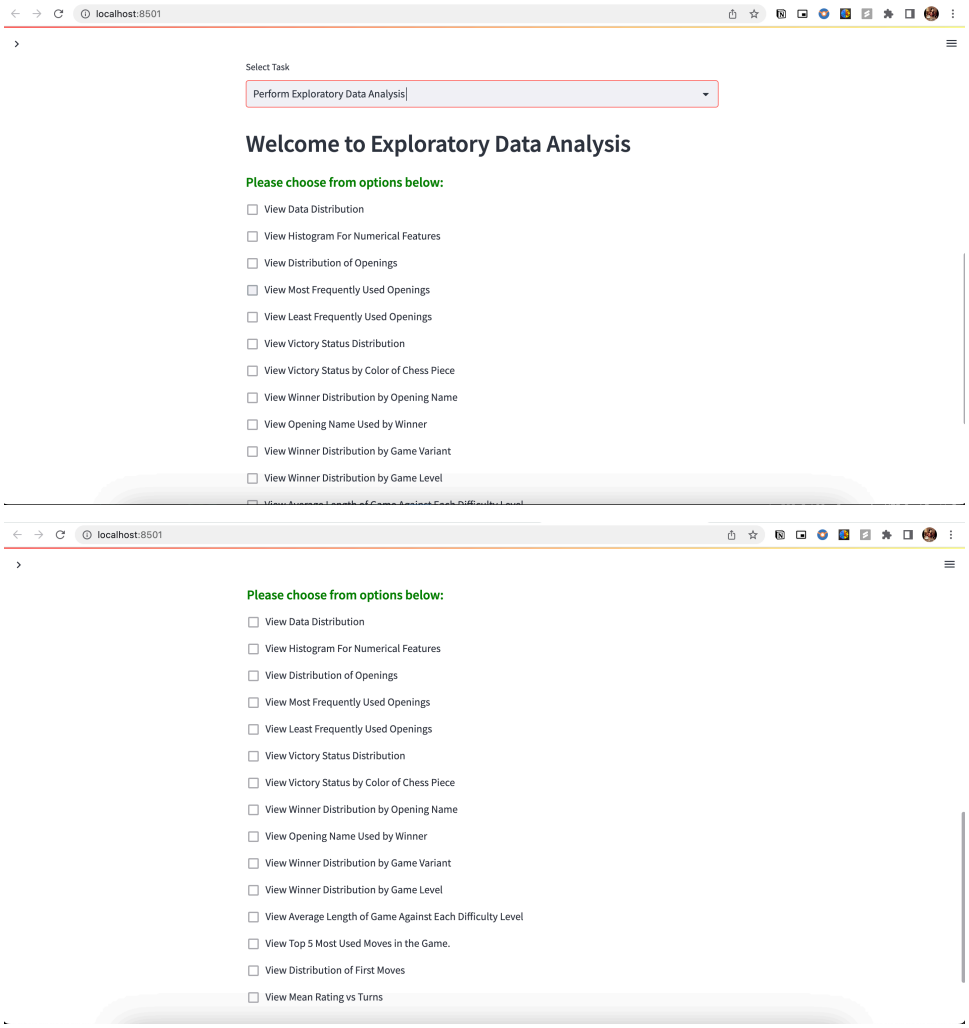

localhost:8501
Statistics of data:
created_at last_move_at turns white_rating black_rating opening_ply
count 20,058 20,058 20,058 20,058 20,058 20,058
mean 1,483,616,852,629.0923 1,483,617,722,336.142 60.466 1,596.6319 1,588.832 4.817
std 28,501,509,421.0049 28,501,400,588.8901 33.5706 291.2534 291.0361 2.7972
min 1,376,771,633,173 1,376,771,863,841 1 784 789 1
25% 1,477,547,500,000 1,477,547,500,000 37 1,398 1,391 3
50% 1,496,010,000,000 1,496,010,000,000 55 1,567 1,562 4
75% 1,503,170,000,000 1,503,170,000,000 79 1,793 1,784 6
max 1,504,493,143,790 1,504,493,827,262 349 2,700 2,723 28
To view first 10 rows of data, click below.
View
To view last 10 rows of data, click below.
View


localhost:8501
To view first 10 rows of data, click below.
View
id rated created_at last_move_at turns victory_status winner incre
0 TZJHLljE 1,504,210,000,000 1,504,210,000,000 13 outoftime white 15+2
1 l1NXvwaE 1,504,130,000,000 1,504,130,000,000 16 resign black 5+10
2 mIICvQHh 1,504,130,000,000 1,504,130,000,000 61 mate white 5+10
3 kWKvrqYL 1,504,110,000,000 1,504,110,000,000 61 mate white 20+0
4 9tXo1AUZ 1,504,030,000,000 1,504,030,000,000 95 mate white 30+3
5 MsoDV9wj 1,504,240,000,000 1,504,240,000,000 5 draw draw 10+0
6 qwU9rasv 1,504,230,000,000 1,504,230,000,000 33 resign white 10+0
7 RVN0N3VK 1,503,680,000,000 1,503,680,000,000 9 resign black 15+3
8 dwF3DJHO 1,503,510,000,000 1,503,510,000,000 66 resign black 15+0
9 afoMwnLg 1,503,440,000,000 1,503,440,000,000 119 mate white 10+0
To view last 10 rows of data, click below.
View

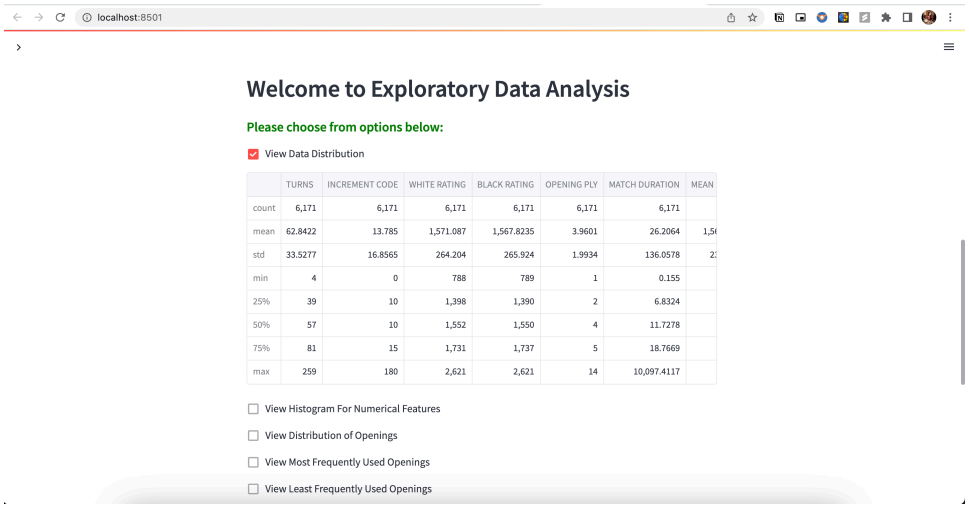

localhost:8501
To view last 10 rows of data, click below.
View
id rated created_at last_move_at turns victory_status winner incre
20,048 dnexZDsv 1,499,869,487,435 1,499,869,745,122 25 resign white 10+1
20,049 7IENcPg3 1,499,814,832,472 1,499,815,248,818 43 mate white 10+0
20,050 nYOvevdh 1,499,814,002,224 1,499,814,052,544 9 outoftime white 10+0
20,051 uMzb0TPC 1,499,812,466,451 1,499,813,212,945 58 mate black 10+1
20,052 EopEqqAa 1,499,811,847,779 1,499,812,436,546 37 resign white 10+1
20,053 EfqH7VVH 1,499,790,914,342 1,499,791,236,076 24 resign white 10+1
20,054 WSJDhbPl 1,499,698,089,760 1,499,698,833,979 82 mate black 10+0
20,055 yrAas0Kj 1,499,697,877,493 1,499,698,050,327 35 mate white 10+0
20,056 b0v4tRyF 1,499,696,127,019 1,499,697,073,718 109 resign white 10+0
20,057 N8G2JHGG 1,499,643,152,649 1,499,643,889,348 78 mate black 10+0

Exploratory Data Analysis:

The user can perform exploratory data analysis by choosing from the list of options provided.



View Data Distribution: This shows the detailed distribution of the uploaded data. The details of data distribution can help the user to understand the statistical representation of the data. Through this process the mean, standard deviation, data count for each column etc. can be accessed. This gives a better idea about the maximum and minimum value present in each column.



View Histogram for Numerical Features: This plots histogram corresponding to TURNS, INCREMENT CODE, WHITE RATING, BLACK RATING, OPEN PLY, and MEAN RATING were plotted to help the user view the distribution of numerical features.

TURNS**:** The **x-axis** represents the range of turns divided into equally spaced bins and the **y-axis** represents the number of observation or the frequency of the turns that fall into each bin.

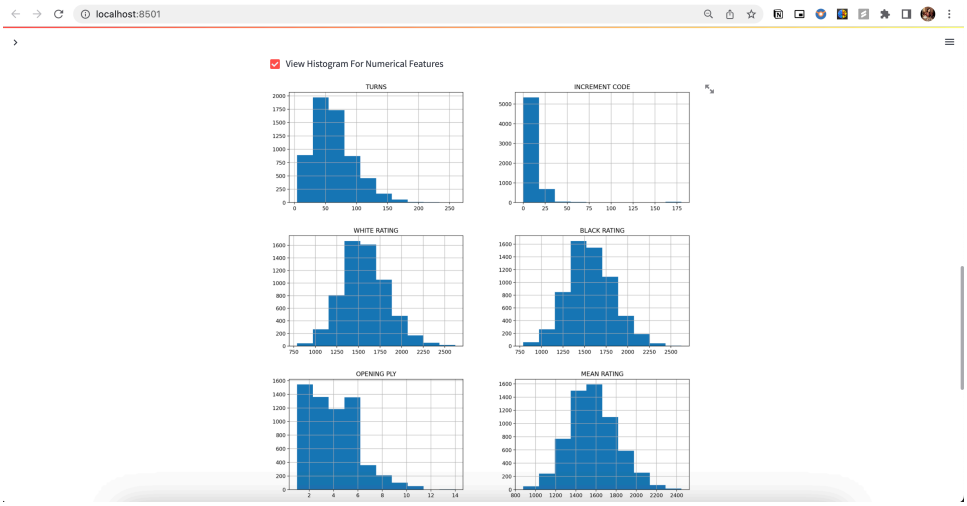
INCREMENT CODE**:** The **x-axis** represents the range of increment code divided into equally spaced bins and the **y-axis** represents the number of observation or the frequency of the increment code that fall into each bin.

WHITE RATING**:** The **x-axis** represents the range of white rating divided into equally spaced bins and the **y-axis** represents the number of observation or the frequency of the white rating that fall into each bin.

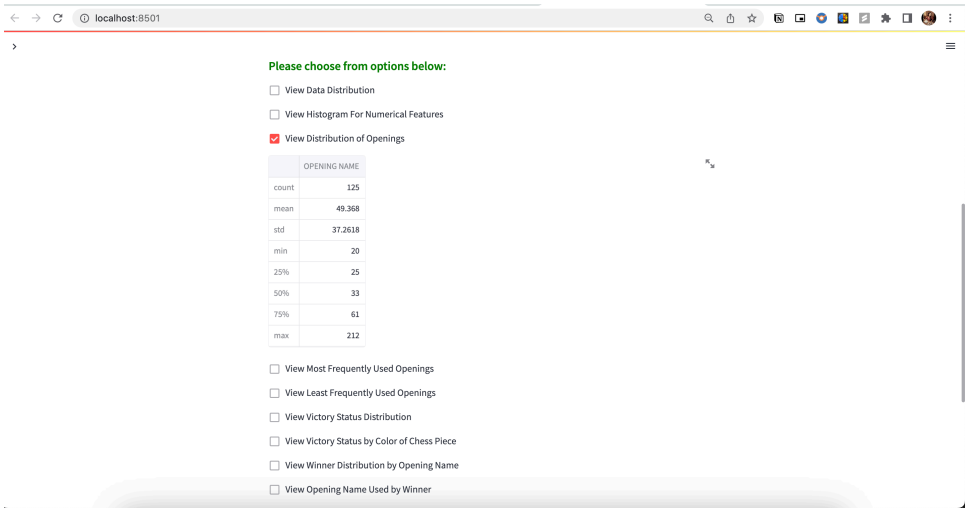
BLACK RATING**:** The **x-axis** represents the range of black rating divided into equally spaced bins and the **y-axis** represents the number of observation or the frequency of the black rating that fall into each bin.

OPEN PLY**:** The **x-axis** represents the range of open ply divided into equally spaced bins and the **y-axis** represents the number of observation or the frequency of the open ply that fall into each bin.

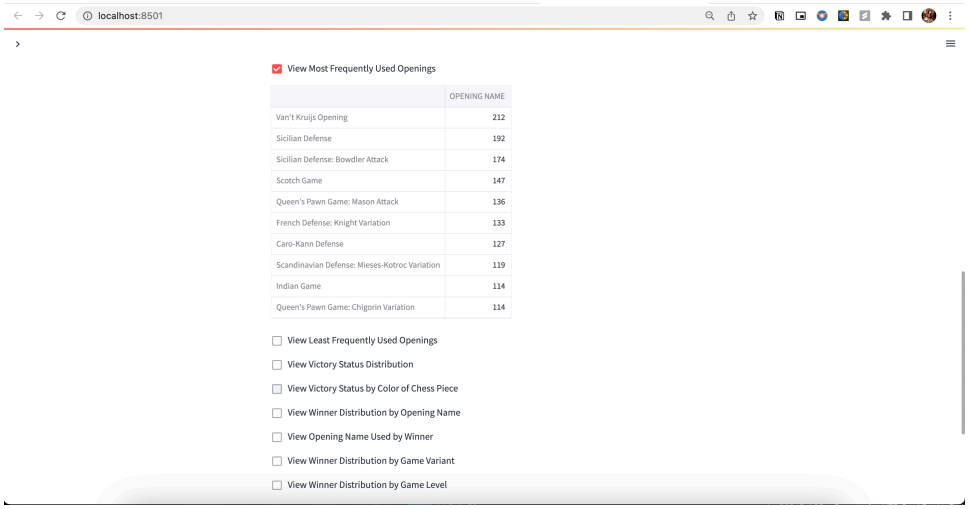
MEAN RATING**:** The **x-axis** represents the range of mean rating divided into equally spaced bins and the **y-axis** represents the number of observation or the frequency of the mean rating that fall into each bin.



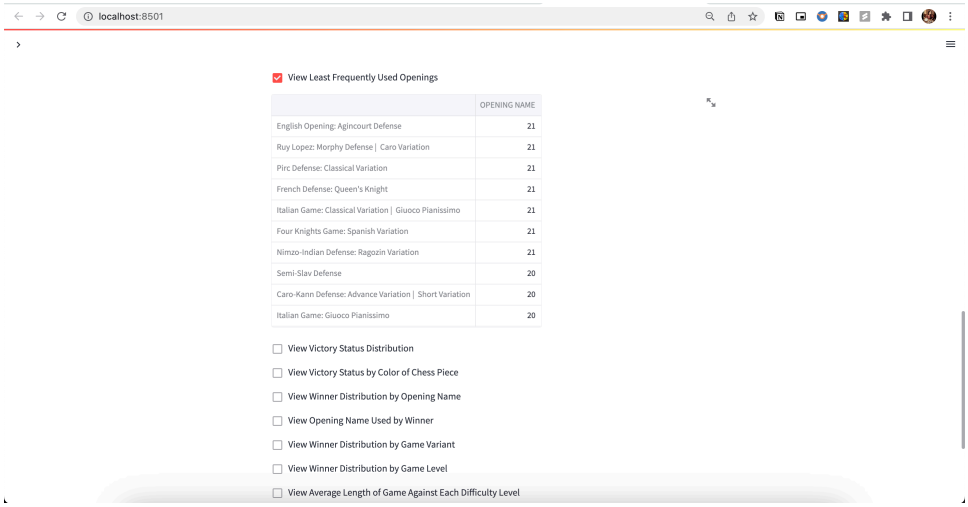
View Distribution of Openings: This shows how the openings are distributed across games. This will help the user understand how the openings are used by chess players. For example, on exploring the distribution of the OPENING NAME feature, the user can understand that the min and 25% of the openings were used only once.



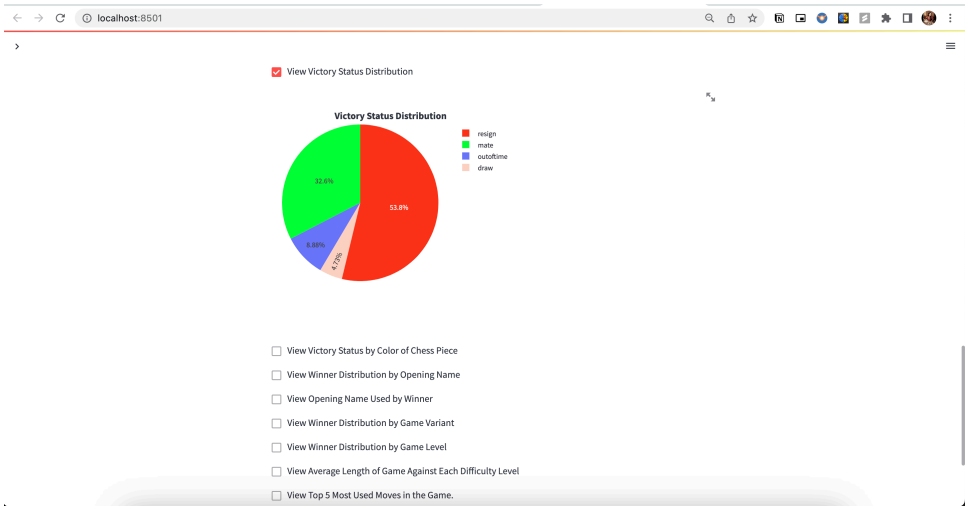
View Most Frequently Used Openings: This shows the most frequently used openings by chess players. This helps the user understand the chess openings preferred by chess players and their frequencies.



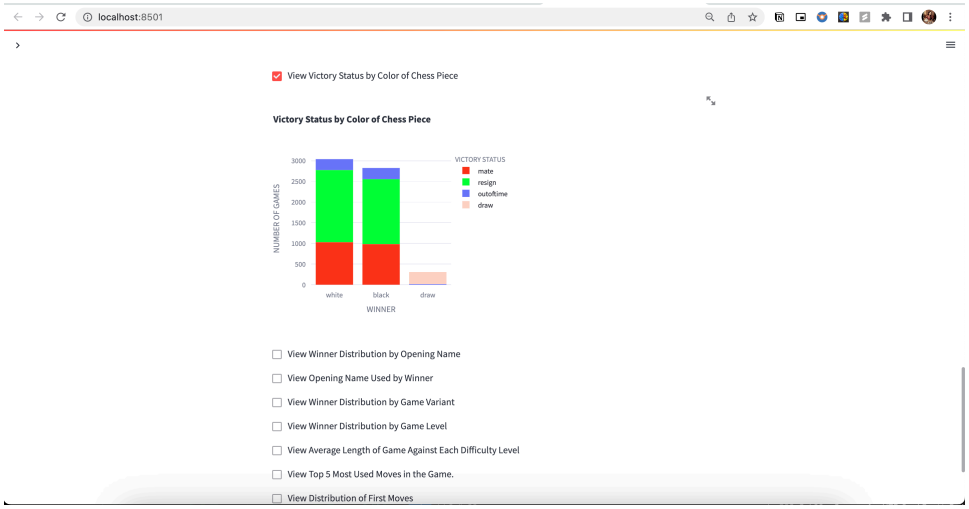
View Least Frequently Used Opening: This shows the least frequently used openings by chess players. This helps the user understand the chess openings not preferred by chess players and their frequencies.



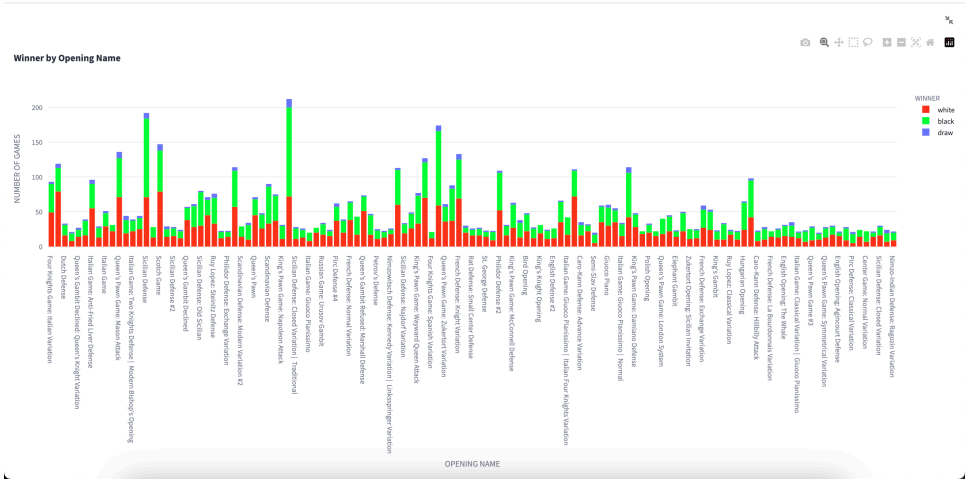
View Victory Status by Distribution: This shows the victory status by distribution. This can help user extract information such as : by analyzing the below graph it can be seen that most of the games end with a player resigning.



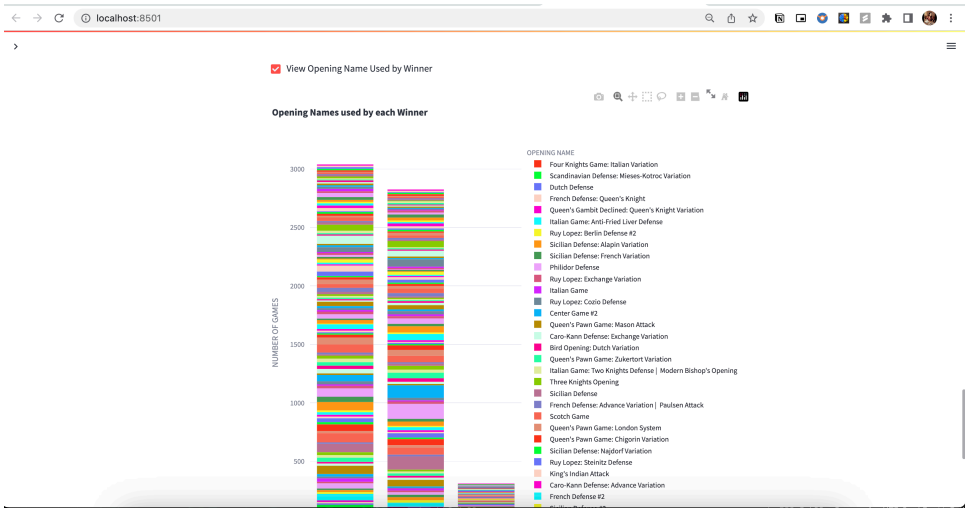
View Victory Status by Color of Chess Piece: This shows the distribution of victory status by color of chess piece. By analyzing this plot the user can infer that whites win more than black.

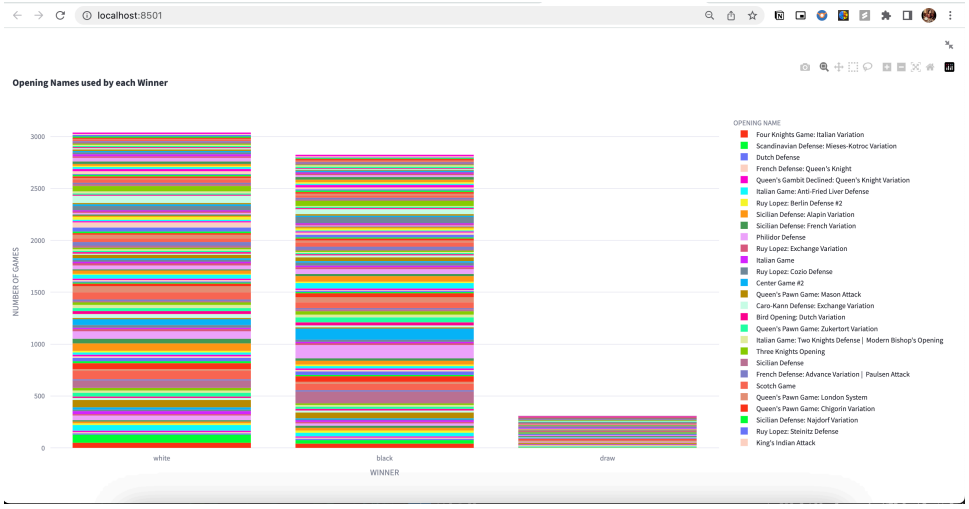


Winner by Opening Names: This shows the distribution of winner by opening names. The user can extract information such: 'Van't Kruji's Opening' is the opening that won the most games.
Black Winner = 128
White Winner = 72
Total wins by using this opening = 200

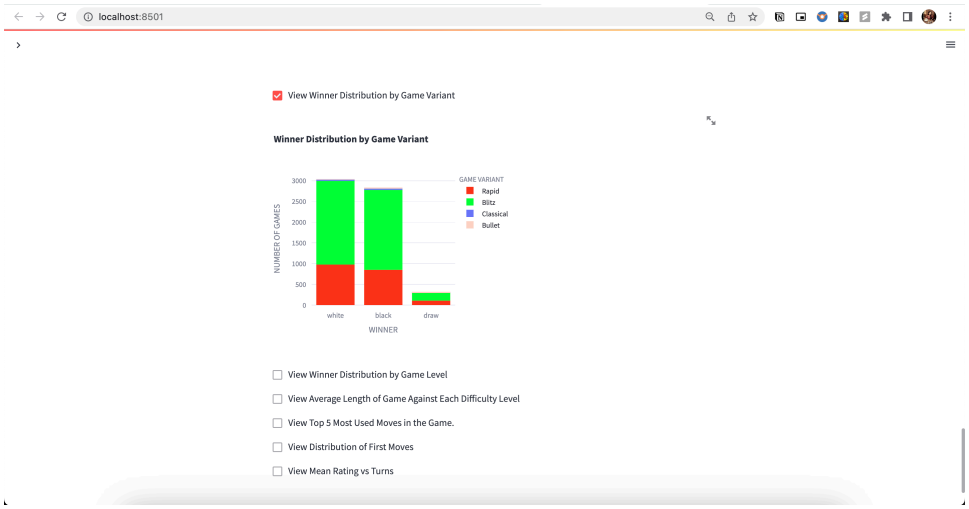


View Opening Name used by Winner: This shows the opening names used by winner and their distribution. By analyzing the plot user can be observe that whites won most using the ‘Scandinavian Defense: Mieses-Kotroc Variation’ opening (count = 79) whereas blacks won most by using the ‘Van’t Kruji’s Opening’ (count = 128).

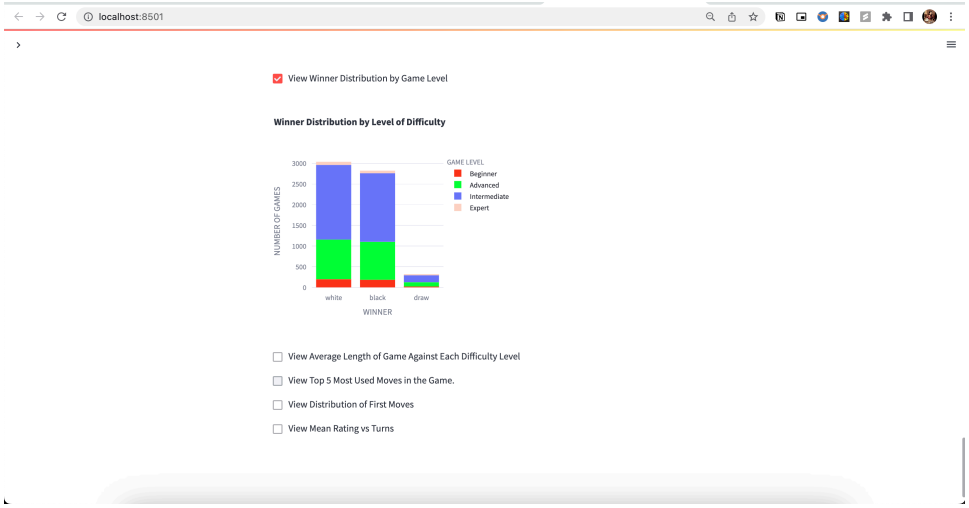




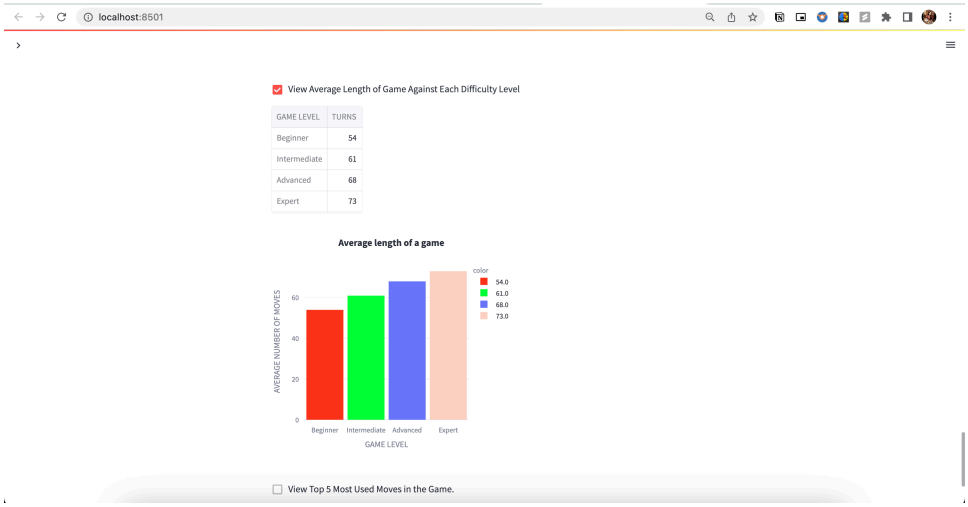
Winner Distribution by Game Variant: This shows the winner distribution by game variant. The user can observe from the below plot that whites won mostly on Blitz(count= 2527) while blacks won on Bullet (count = 21).



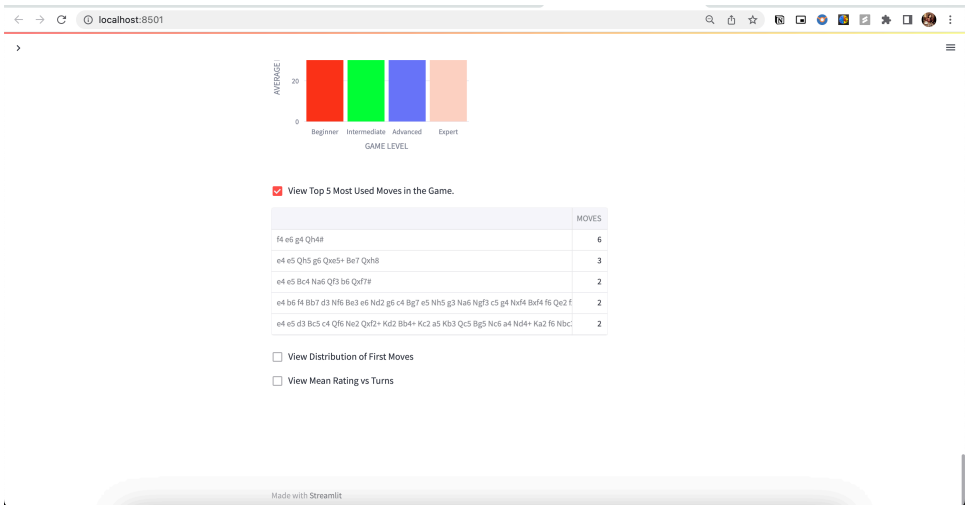
Winner Distribution by Level of Difficulty: This shows the distribution of winners by the level of difficulty of the game. From the plot the user can infer that whites won on every level of difficulty



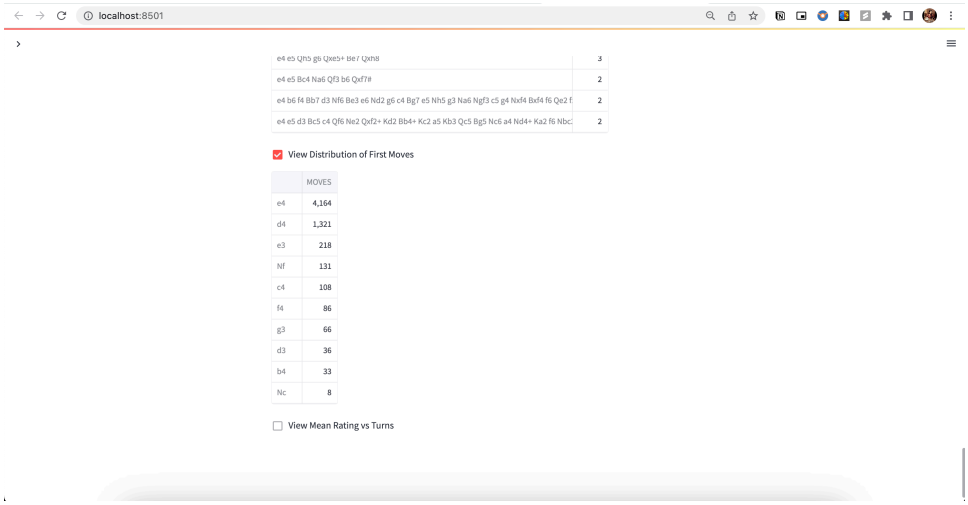
View Average Length of Game against each Difficulty Level: This can help the user understand information such as that with the increase of difficulty level, the average length of the match also increases.



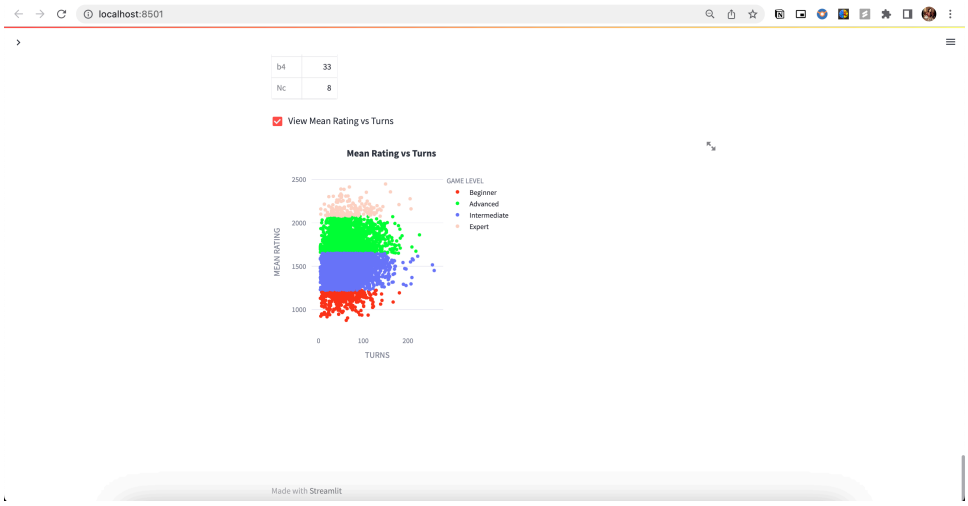
View Top 5 Most Used Moves in the Game: From this the user can infer the most used moves used by players in the game of chess.



View Distribution of First Moves: The user can understand the preferred and not so preferred first moves of chess players through this.



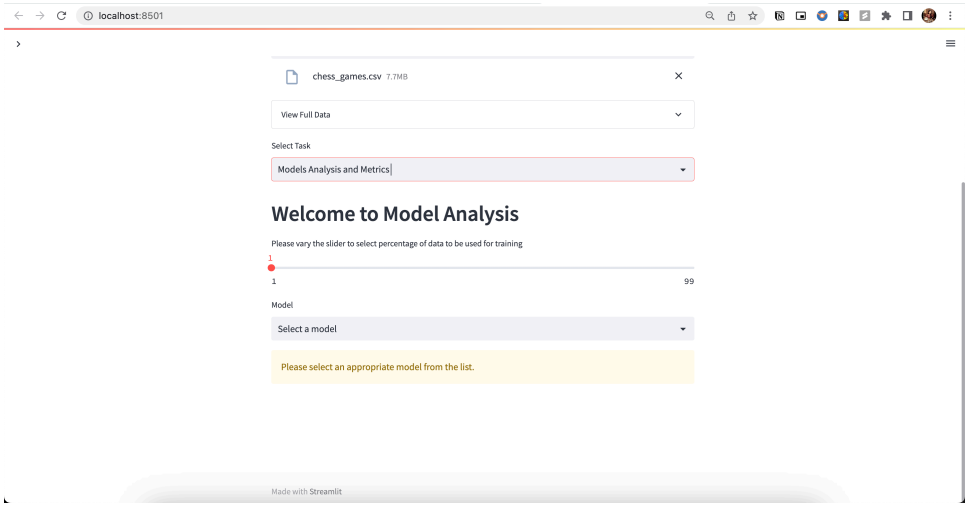
View Mean Rating vs Turns: The user can understand how mean rating relates to the number of turns by this visualization. For example, through the plot below, the user can understand that with the increase of mean rating, the number of turns used in the match also increases.



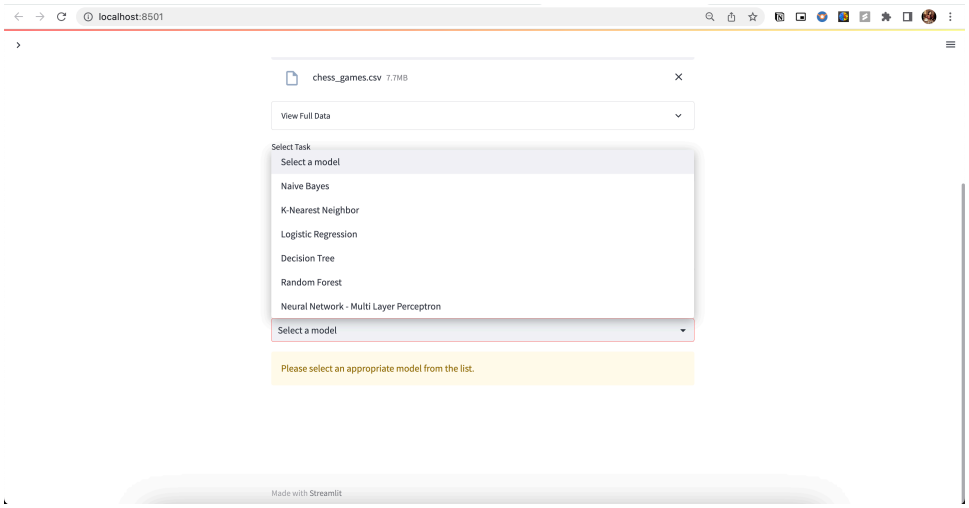
Model Analysis and Metrics:

Through this the user can choose how the training and the test data will be split and can select the model to train, test and evaluate.

The slider can be adjusted to adjust the percentage of train and test data.



The user can select a model from the list.



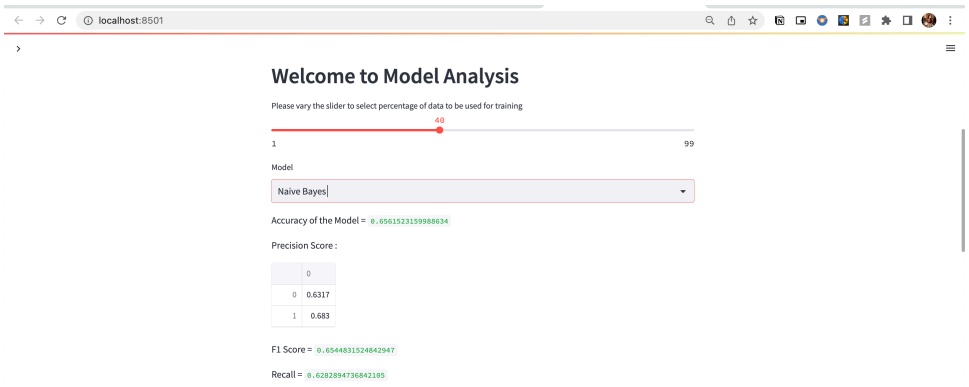
Naïve Bayes Model: The training data size is chosen as 40. Evaluation metrics used are Accuracy of the model, Precision score, F1 score and Recall.

Accuracy: The model made correct predictions for approximately 65.61% of the total instances it predicted.

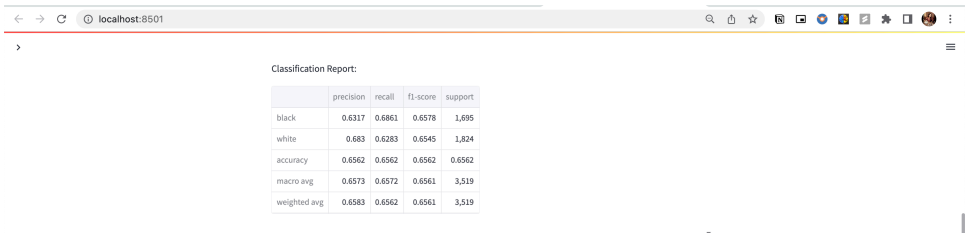
Precision: For Black (0) the model, approximately 63.17% of the predicted instances were actually black. For White (1), 68.3% of the predicted instances were actually white.

F1-Score: The model achieved an overall balanced performance between precision and recall, resulting in a score of 65.44%.

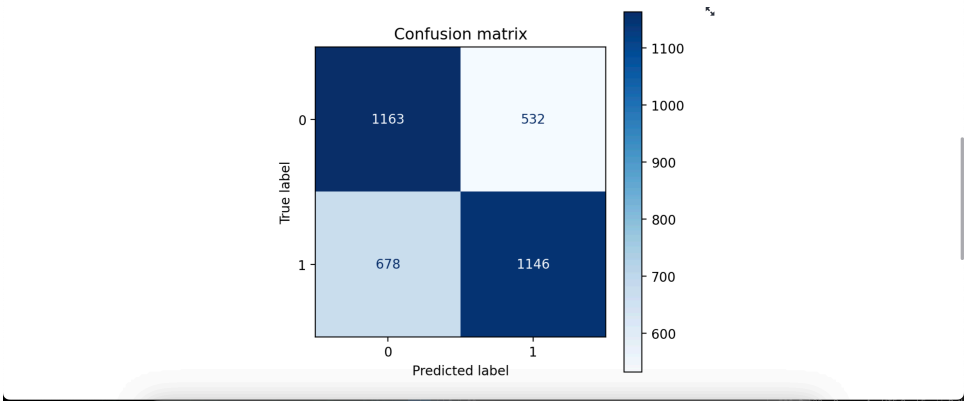
Recall: The model successfully identified approximately 62.82% of the actual positive instances in the dataset.



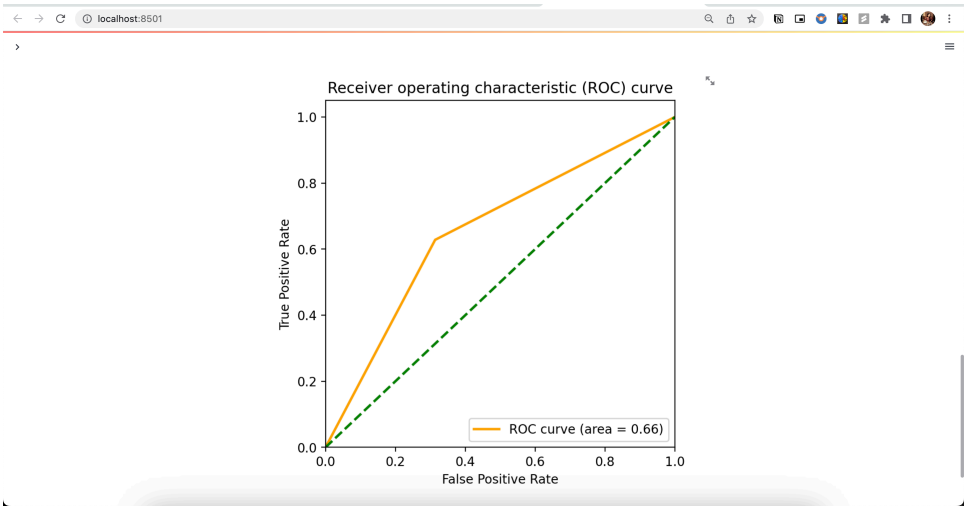
Classification Report Details:



Confusion Matrix: The model correctly classified 1163 records as positive and 1146 records as negative. The model incorrectly classified 678 records as negative and 532 records as positive.



ROC Curve: This model is correctly identifying the positive cases 66% of the time and incorrectly identifies the negative cases as positive 34% of the time.



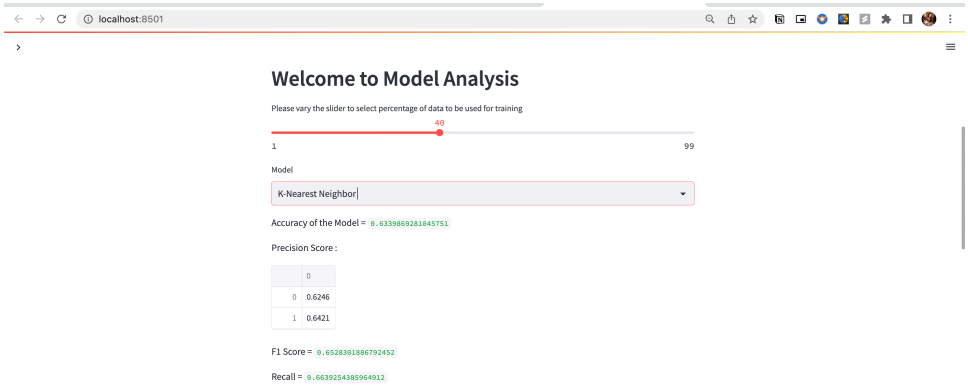
K-Nearest Neighbor: The training data size is chosen as 40. Evaluation metrics used are Accuracy of the model, Precision score, F1 score and Recall.

Accuracy: The model made correct predictions for approximately 63.39% of the total instances it predicted.

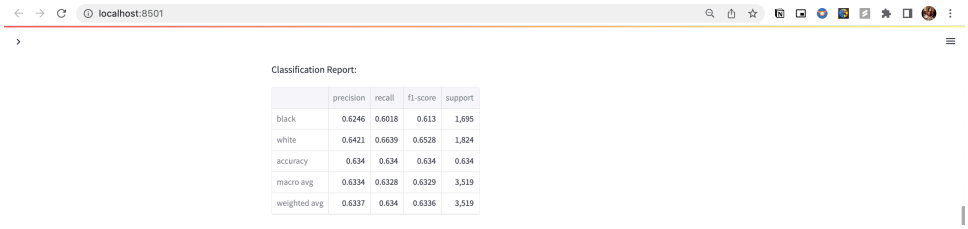
Precision: For Black (0) the model, approximately 62.46% of the predicted instances were actually black. For White (1), 64.21% of the predicted instances were actually white.

F1-Score: The model achieved an overall balanced performance between precision and recall, resulting in a score of 65.28%.

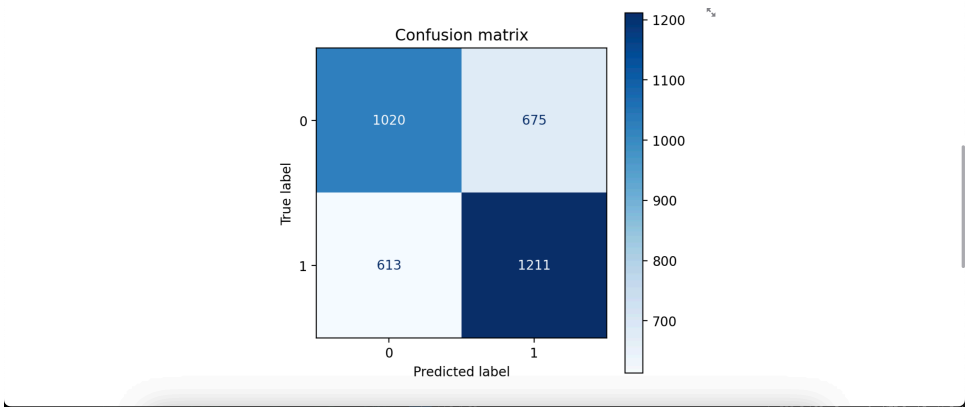
Recall: The model successfully identified approximately 66.39% of the actual positive instances in the dataset.



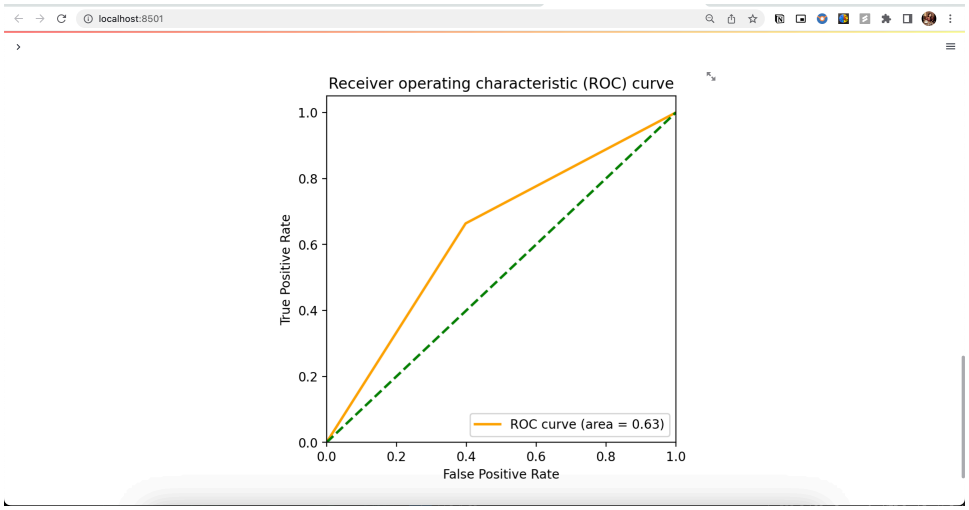
Classification Report Details:



Confusion Matrix: The model correctly classified 1020 records as positive and 1211 records as negative. The model incorrectly classified 613 records as negative and 675 records as positive.



ROC Curve: This model is correctly identifying the positive cases 63% of the time and incorrectly identifies the negative cases as positive 37% of the time.



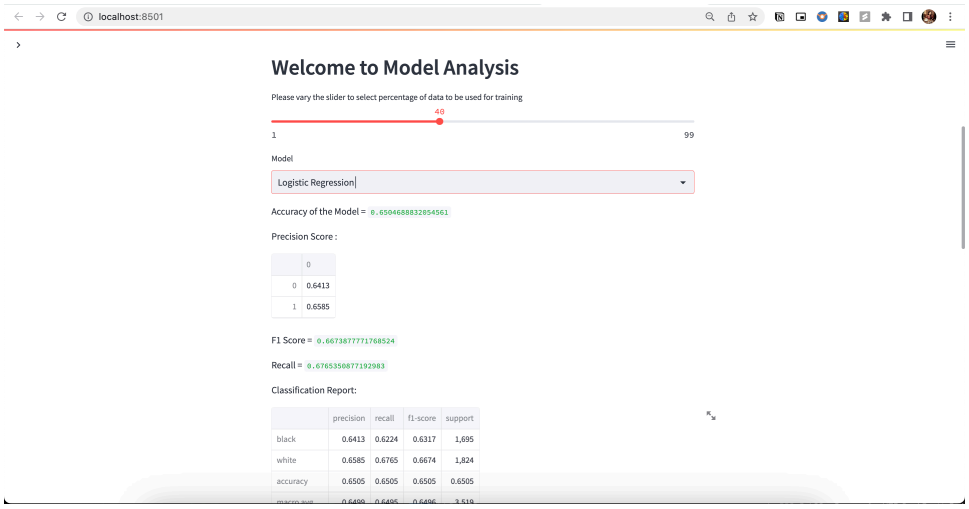
Logistic Regression: The training data size is chosen as 40. Evaluation metrics used are Accuracy of the model, Precision score, F1 score and Recall.

Accuracy: The model made correct predictions for approximately 65.04% of the total instances it predicted.

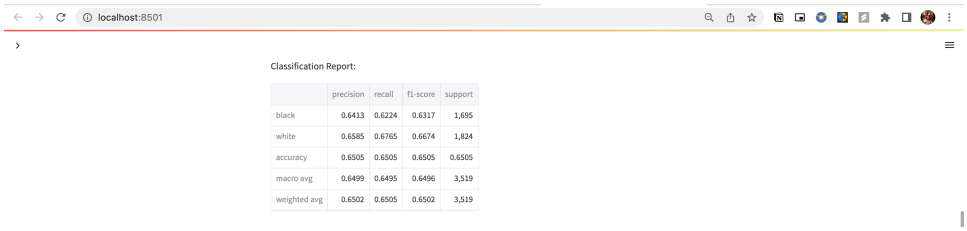
Precision: For Black (0) the model, approximately 64.13% of the predicted instances were actually black. For White (1), 65.85% of the predicted instances were actually white.

F1-Score: The model achieved an overall balanced performance between precision and recall, resulting in a score of 66.73%.

Recall: The model successfully identified approximately 67.66% of the actual positive instances in the dataset.

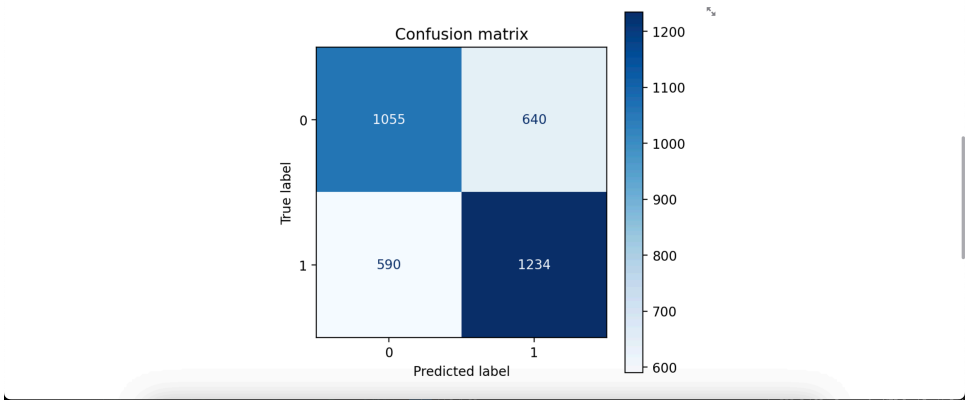


Classification Report Details:

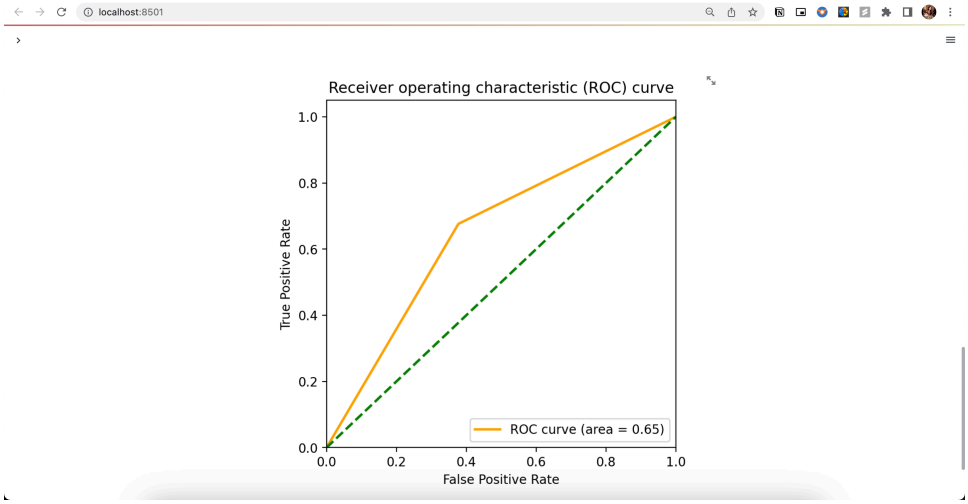


	precision	recall	f1-score	support
black	0.6413	0.6224	0.6317	1,695
white	0.6585	0.6765	0.6674	1,824
accuracy	0.6505	0.6505	0.6505	0.6505
macro avg	0.6499	0.6495	0.6496	3,519
weighted avg	0.6502	0.6505	0.6502	3,519

Confusion Matrix: The model correctly classified 1055 records as positive and 1234 records as negative. The model incorrectly classified 590 records as negative and 640 records as positive.



ROC Curve: This model is correctly identifying the positive cases 65% of the time and incorrectly identifies the negative cases as positive 35% of the time.



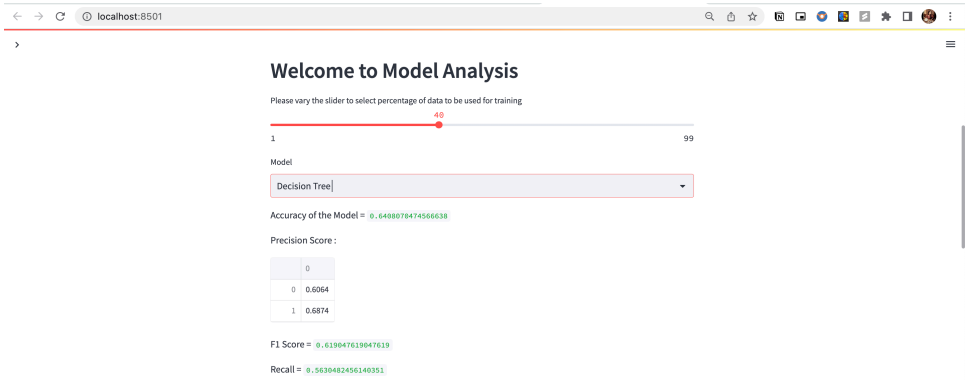
Decision Tree: The training data size is chosen as 40. Evaluation metrics used are Accuracy of the model, Precision score, F1 score and Recall.

Accuracy: The model made correct predictions for approximately 64.08% of the total instances it predicted.

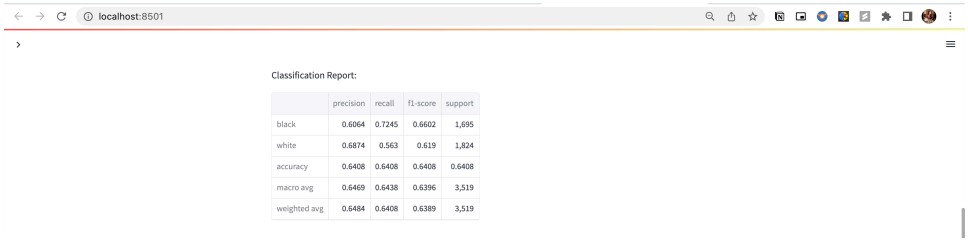
Precision: For Black (0) the model, approximately 60.64% of the predicted instances were actually black. For White (1), 68.74% of the predicted instances were actually white.

F1-Score: The model achieved an overall balanced performance between precision and recall, resulting in a score of 61.90%.

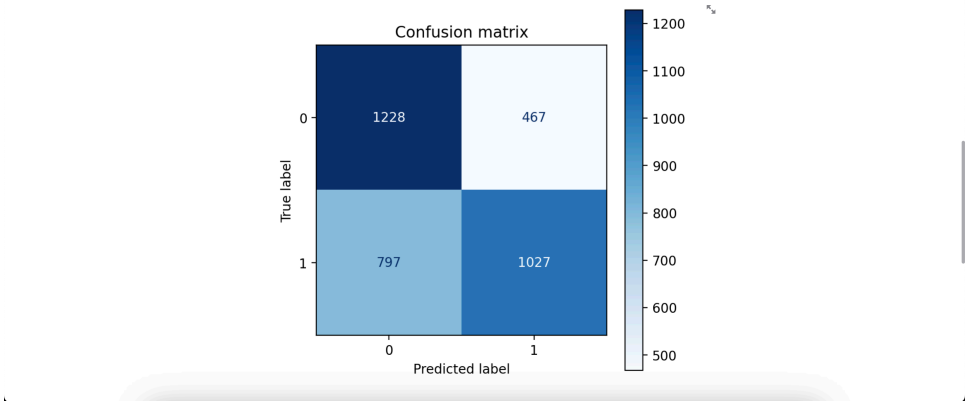
Recall: The model successfully identified approximately 56.30% of the actual positive instances in the dataset.



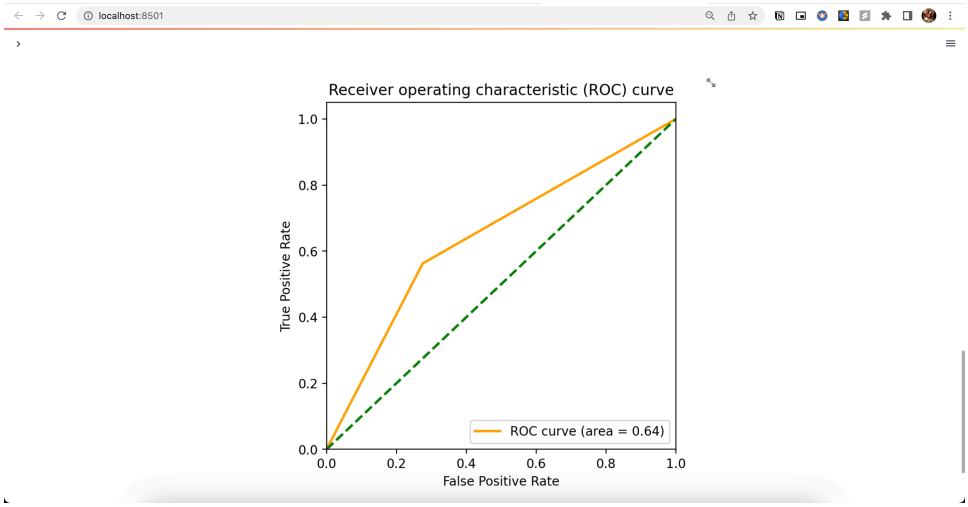
Classification Report Details:



Confusion Matrix: The model correctly classified 1228 records as positive and 1027 records as negative. The model incorrectly classified 797 records as negative and 467 records as positive.



ROC Curve: This model is correctly identifying the positive cases 64% of the time and incorrectly identifies the negative cases as positive 36% of the time.



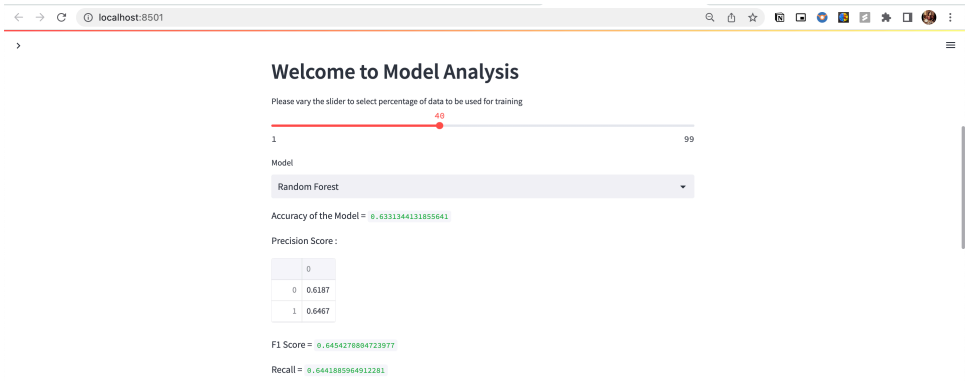
Random Forest: The training data size is chosen as 40. Evaluation metrics used are Accuracy of the model, Precision score, F1 score and Recall.

Accuracy: The model made correct predictions for approximately 63.31% of the total instances it predicted.

Precision: For Black (0) the model, approximately 61.87% of the predicted instances were actually black. For White (1), 64.67% of the predicted instances were actually white.

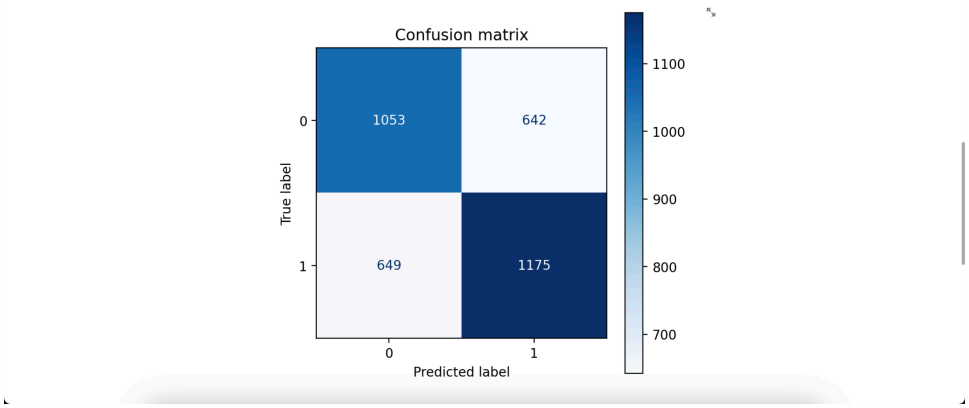
F1-Score: The model achieved an overall balanced performance between precision and recall, resulting in a score of 64.54%.

Recall: The model successfully identified approximately 64.41% of the actual positive instances in the dataset.

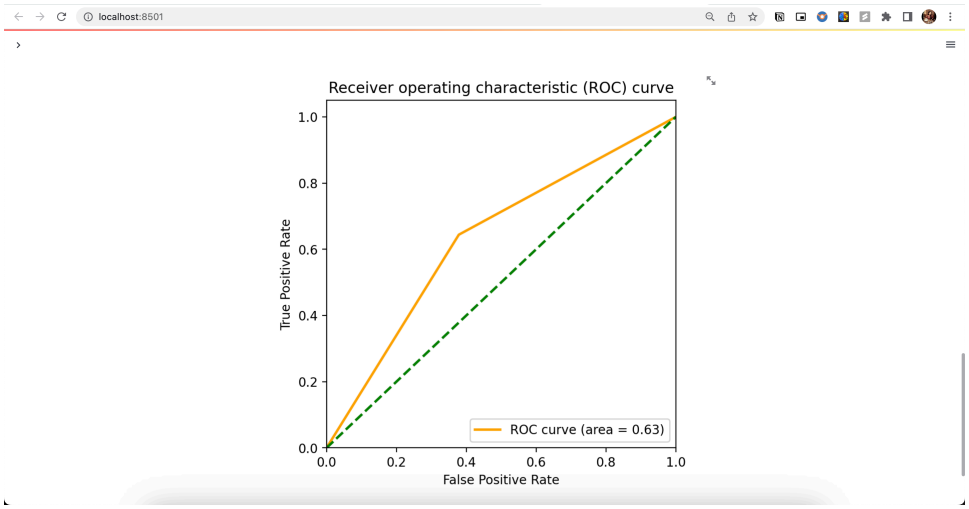


Classification Report Details:

Confusion Matrix: The model correctly classified 1053 records as positive and 1175 records as negative. The model incorrectly classified 649 records as negative and 642 records as positive.



ROC Curve: This model is correctly identifying the positive cases 63% of the time and incorrectly identifies the negative cases as positive 37% of the time.



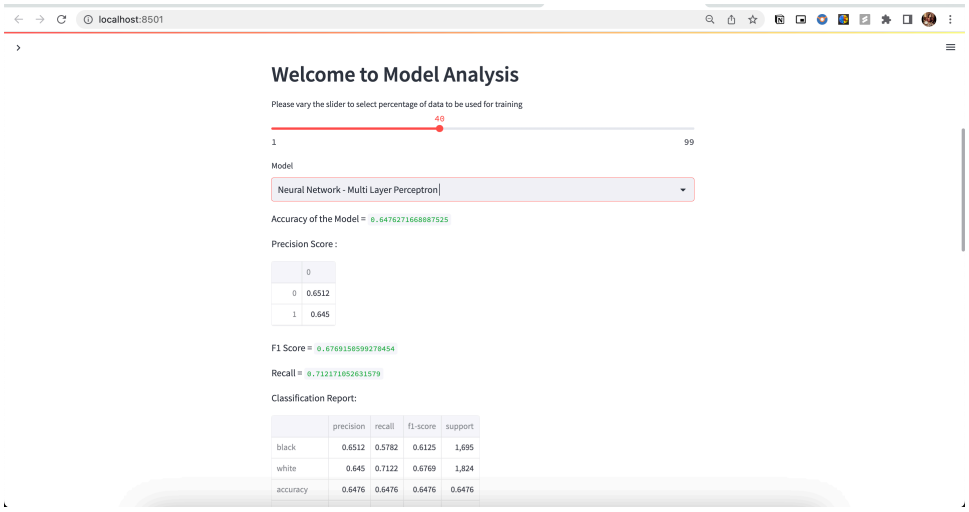
Neural Network – Multi Layer Perceptron: The training data size is chosen as 40. Evaluation metrics used are Accuracy of the model, Precision score, F1 score and Recall.

Accuracy: The model made correct predictions for approximately 64.76% of the total instances it predicted.

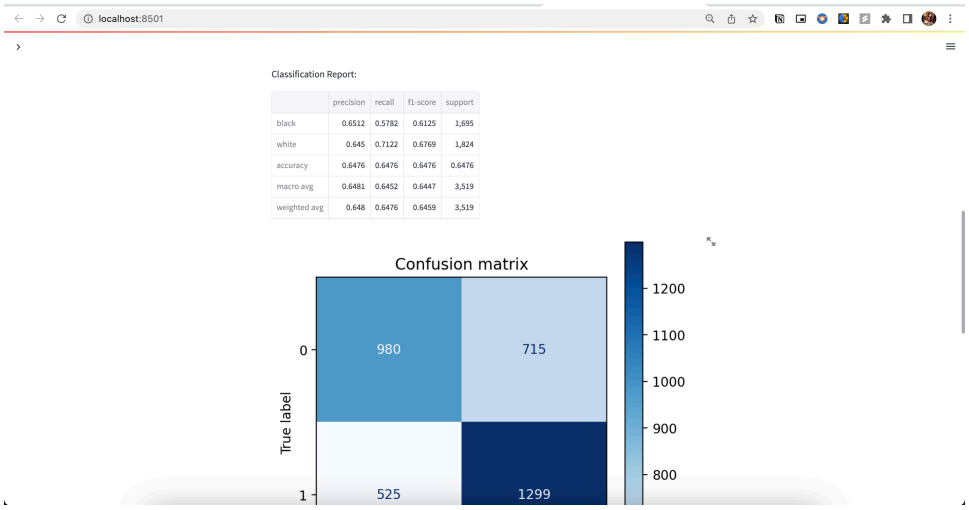
Precision: For Black (0) the model, approximately 65.12% of the predicted instances were actually black. For White (1), 64.5% of the predicted instances were actually white.

F1-Score: The model achieved an overall balanced performance between precision and recall, resulting in a score of 67.69%.

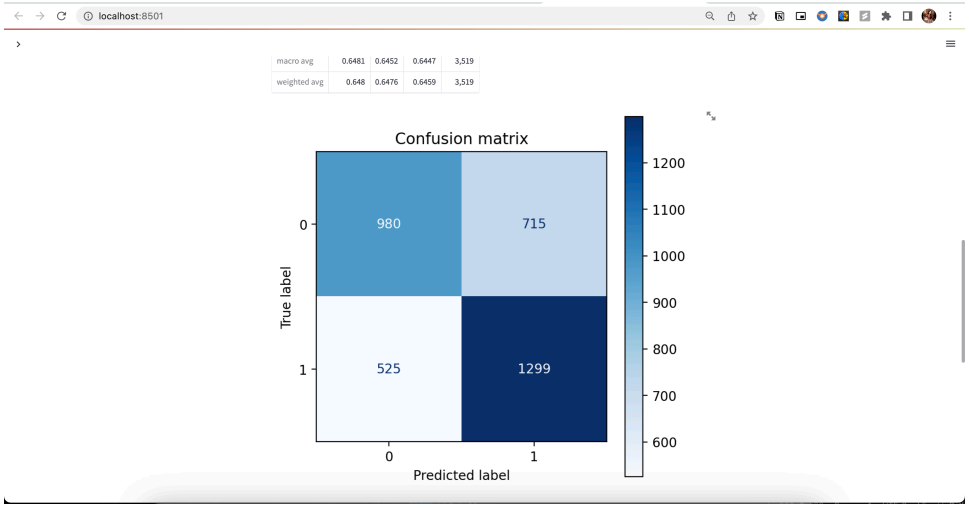
Recall: The model successfully identified approximately 71.21% of the actual positive instances in the dataset.



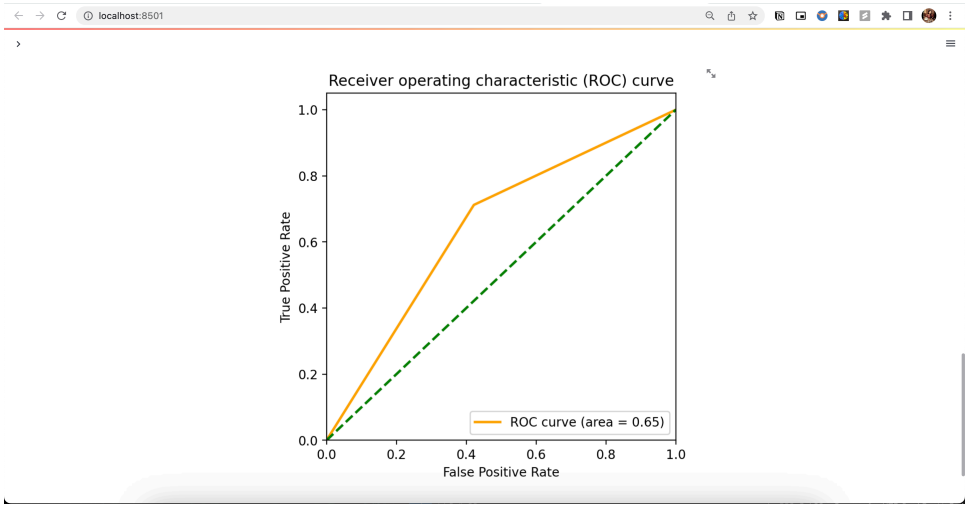
Classification Report Details:



Confusion Matrix: The model correctly classified 980 records as positive and 1299 records as negative. The model incorrectly classified 525 records as negative and 715 records as positive.

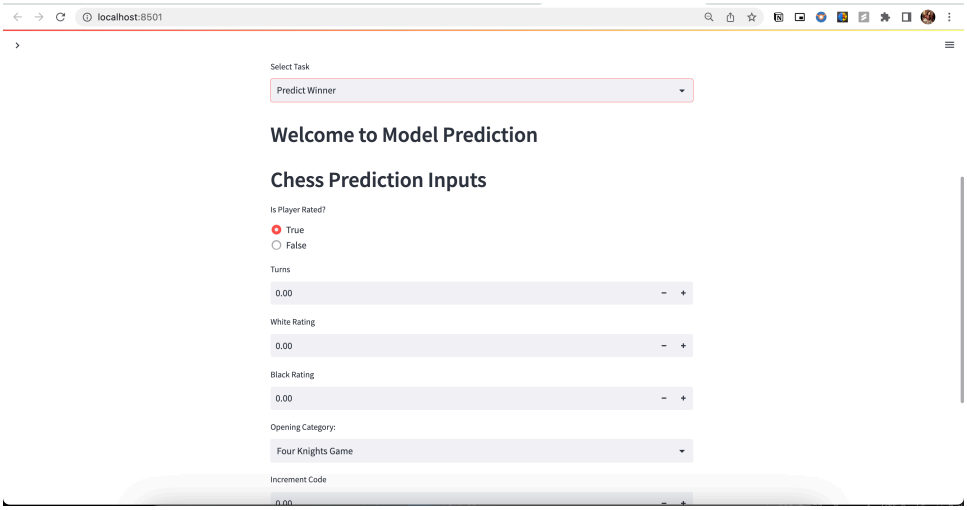


ROC Curve: This model is correctly identifying the positive cases 65% of the time and incorrectly identifies the negative cases as positive 35% of the time.

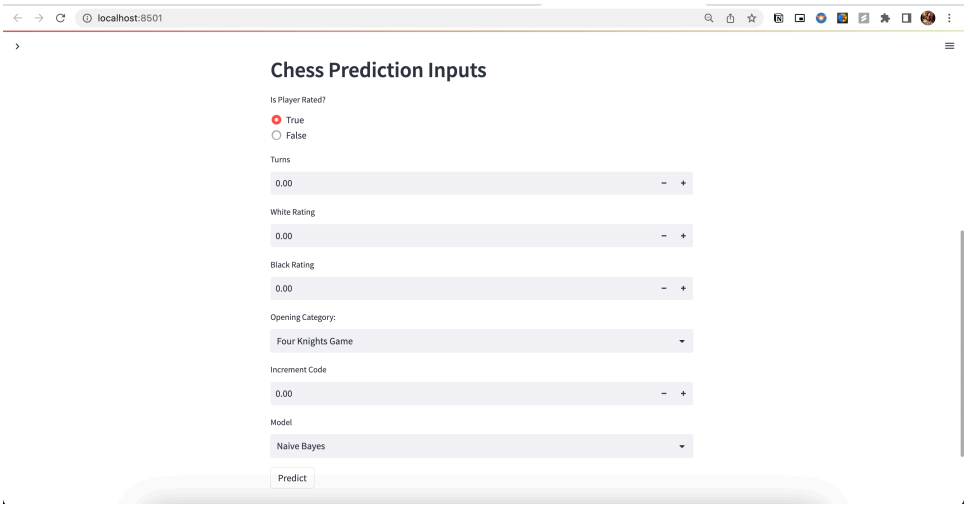


Model Prediction:

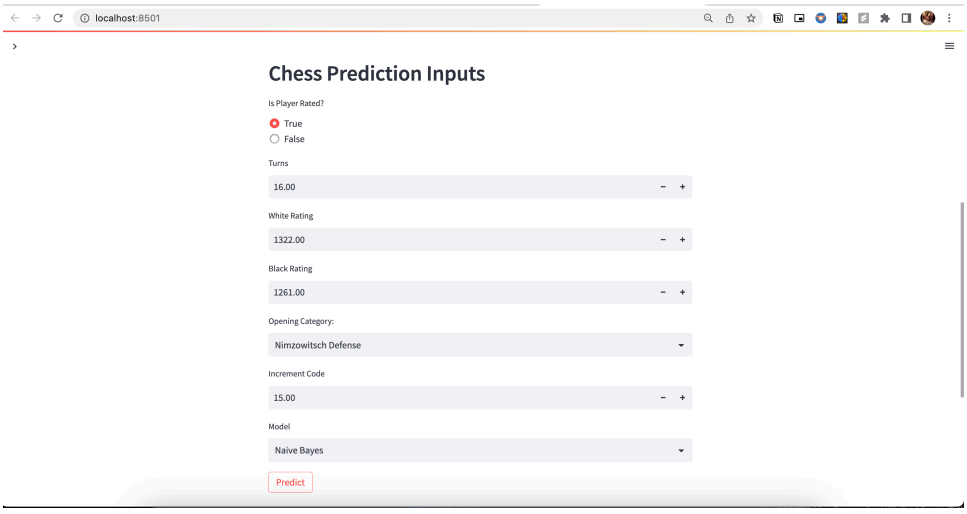
This screen allows a user to predict on a custom data.



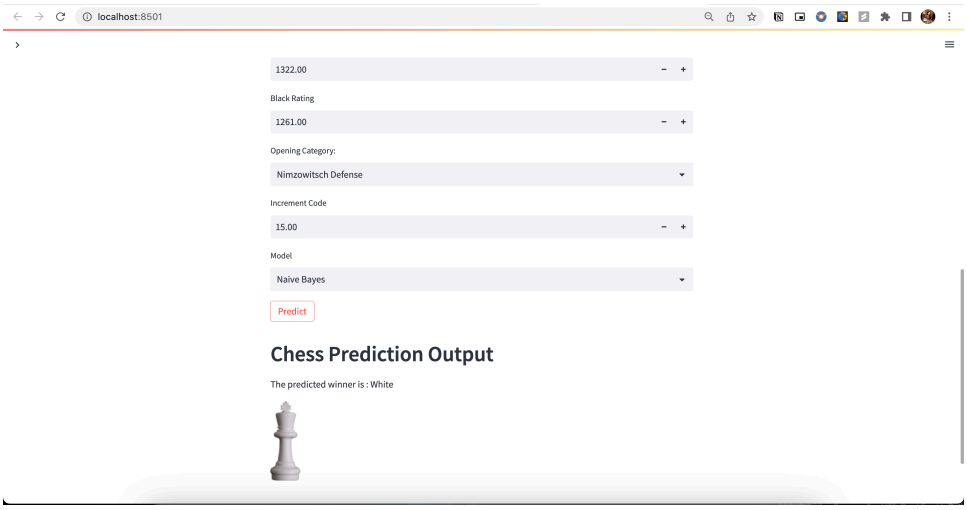
On selecting Predict Winner from the task menu, user is taken to the above screen.



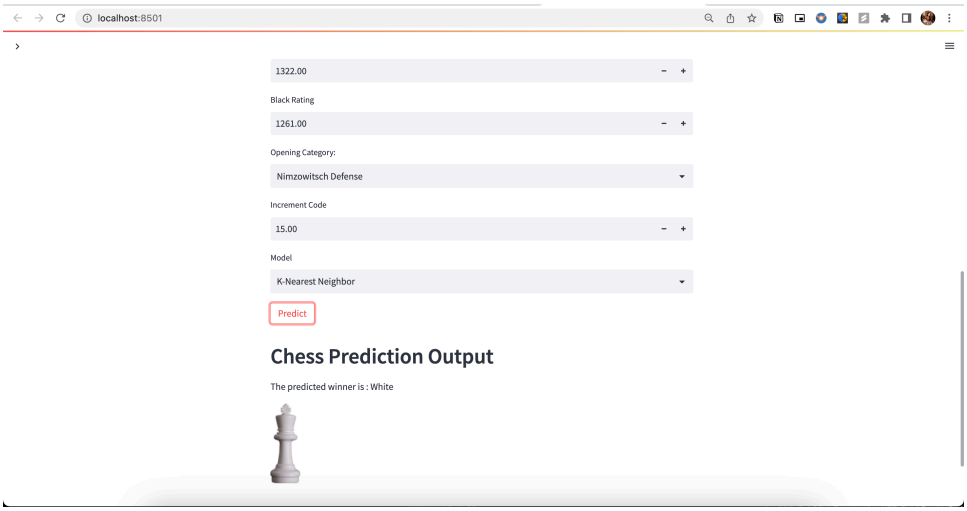
On this screen, the first field is for feature RATED which takes input as True or False using a radio button. There are numeric input fields for features TURNS, WHITE RATING, BLACK RATING, and INCREMENT CODE. Using the values provided for WHITE RATING and BLACK RATING, MEAN RATING feature value is calculated. Next is the dropdown menu for selecting the OPENING CATEGORY which is populated using the extracted information from uploaded dataset.



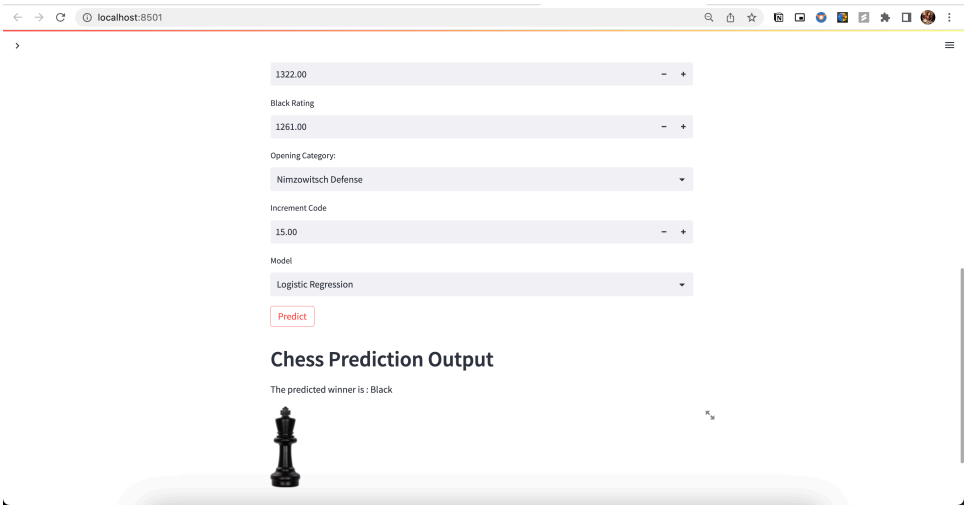
After providing all the values, user can select one of the Models from the dropdown menu named Model and click Predict Button.



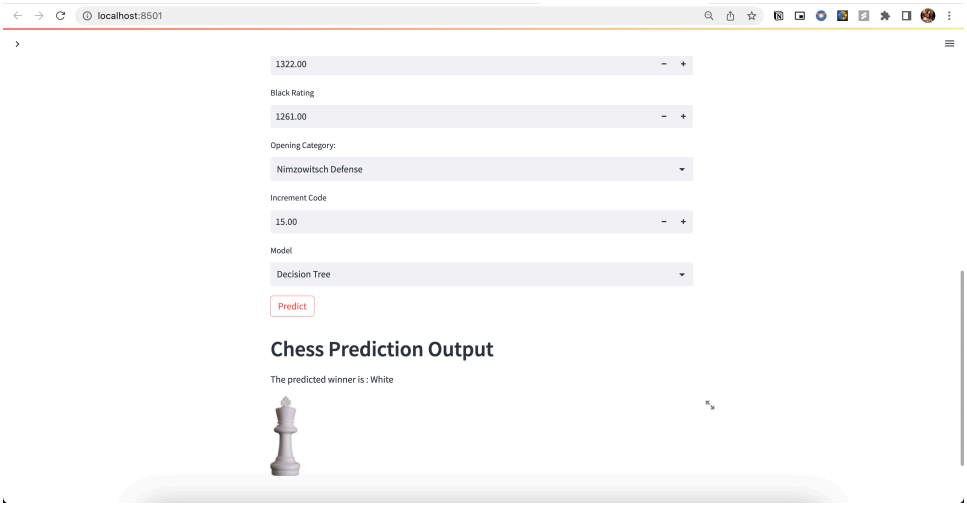
As soon as 'Predict' button is clicked, it will initiate the training of model in backend and then pass the input values to obtain a prediction from the selected model. In the screenshot above, the prediction is generated by Naïve Bayes Model.



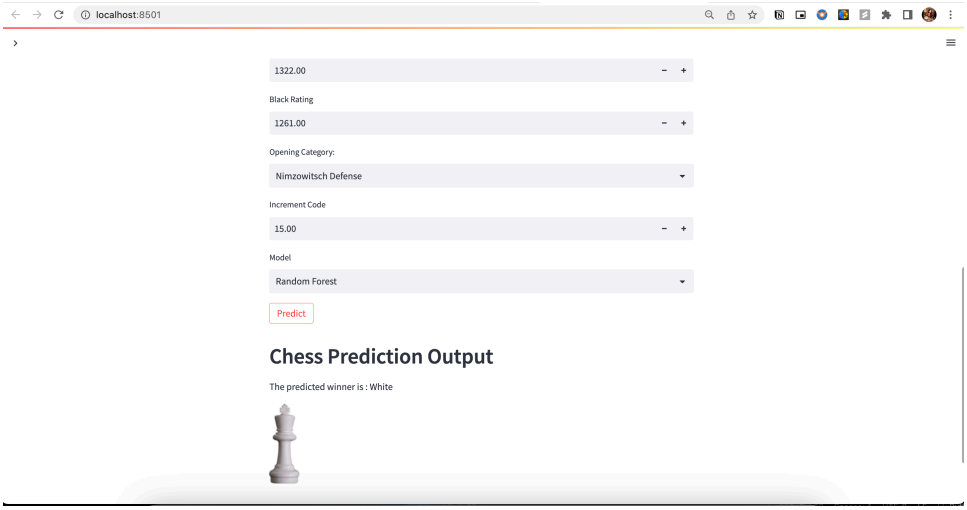
In the screenshot above, the prediction is generated by K-Nearest Neighbor Model.



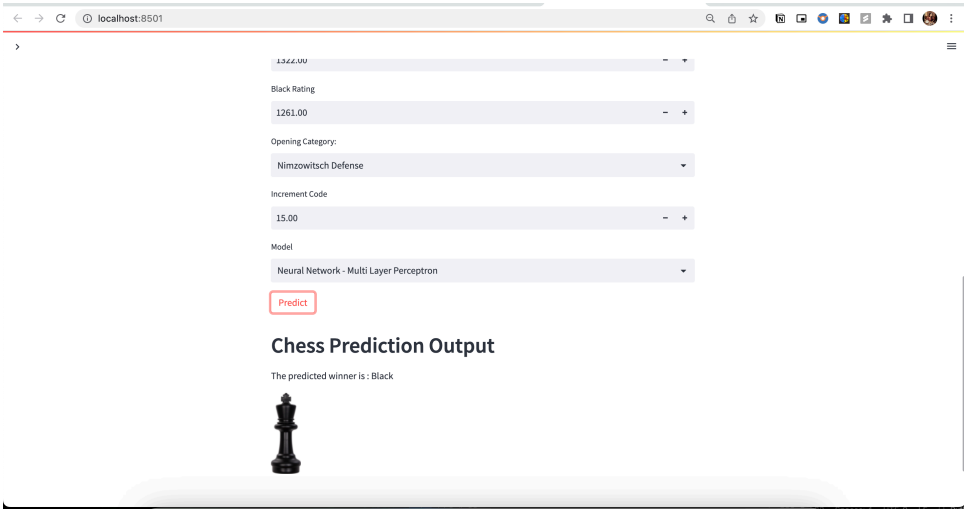
In the screenshot above, the prediction is generated by Logistic Regression Model.



In the screenshot above, the prediction is generated by Decision Tree Model.



In the screenshot above, the prediction is generated by Random Forest Model.



In the screenshot above, the prediction is generated by Neural Network – Multi Layer Perceptron Model.

IX. EXECUTION STEPS

Please follow the below steps for executing steps:
Phase 1:
Load the .ipynb file and execute the code blocks one by one in order.

Phase 2:
Load the .ipynb file and execute the code blocks one by one in order.

Phase 3:
1. Open terminal at phase3 folder
2. run command - “chmod +x run.sh”
3. run command - “bash run.sh”

This will install required libraries, so an active internet connection is required in the executing machine and then it will initialize web application.
After web application is up, the dataset can be found in src/phase3/data/chess_data.csv

CONCLUSION

From Phase 2 of the project, the following can be concluded.

a. All models used performed similar on the Lichess Chess Game Dataset and had comparable accuracy when predicting the winner of a chess game.
b. K-Nearest Neighbor and Decision Tree models integrating GridSearchCV with the base model gave better results.
c. The similarity across the performance, as measured through the evaluation metric, over all models used can be a result of the correlation between the features of the dataset. The co-relation between the features of the dataset us depicted in the co-relation matrix below.

ACKNOWLEDGMENT

We would like to take advantage of this opportunity to express our gratitude to Dr. Eric Mikida and Dr. Shamsad Parvin, our lecturers and Smarana Shrikant Pankati, our assigned project mentor for their insightful comments, patience, and time.

REFERENCES

1. https://bookdown.org/pq2142/finalproj/
2. https://community.ibm.com/community/user/ai-datascience/blogs/moloy-de1/2020/08/27/points-to-ponder
3. https://towardsdatascience.com/analysing-lichess-games-with-r-c4f8b0bc512c
4. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
5. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
6. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
7. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
8. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
9. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
10. https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html - :~:text=The precision-recall curve shows,a low false negative rate.

11. https://docs.streamlit.io/library/get-started