



PROJECT REPORT  
ON  
**CHESS WINNER PREDICTION**

(Information Technology)

*SUBMITTED BY*

Mr. ATHARVA CHAVAN

Mr. ANUJ CHITARI

Mr. SHASHWAT TRIPATHI

Mr. PIYUSH TILOKANI

DEPARTMENT OF INFORMATION TECHNOLOGY  
V.E.S. INSTITUTE OF TECHNOLOGY  
2024-25

# *Certificate*

This is to certify that project entitled

**“ CHESS WINNER PREDICTION MODEL”**

**Group Members Names:**

Mr. ATHARVA CHAVAN

Mr. ANUJ CHITARI

Mr. SHASHWAT TRIPATHI

Mr. PIYUSH TILOKANI

**Mr. JITENDRA MADAVI**

**Project Mentor**

**Dr.(Mrs.)SHALU CHOPRA**  
**H.O.D**

**Dr.(Mrs.)J.M.Nair**  
**Principal**

Date:

Place: VESIT, Chembur

College Seal

## *Declaration*

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

-ATHARVA CHAVAN (10)

-ANUJ CHITARI (11)

-SHASHWAT TRIPATHI(64)

-PIYUSH TILOKANI(63)

## ACKNOWLEDGEMENT

The project report on "Chess Winner Prediction" is the outcome of the guidance, moral support and devotion bestowed on our group throughout our work. For this we acknowledge and express our profound sense of gratitude to everybody who has been the source of inspiration throughout project preparation. First and foremost we offer our sincere phrases of thanks and innate humility to "HOD Dr.(Mrs.)Shalu Chopra", "Deputy HOD Dr.(Mr.)Manoj Sabnis", "Mr. Jitendra Madavi" for providing the valuable inputs and the consistent guidance and support provided by them. We can say in words that we must at outset tender our intimacy for receipt of affectionate care to Vivekanand Education Society's Institute of Technology for providing such a stimulating atmosphere and conducive work environment.

## Abstract

In the world of chess, predicting the outcome of a game has long intrigued enthusiasts, players, and analysts alike. The "Chess Winner Prediction" project aims to leverage machine learning techniques to forecast the likely victor of a chess match based on various features extracted from historical game data. By analyzing past game records, including player ratings, opening moves, and game duration, the project seeks to develop predictive models capable of discerning patterns and tendencies that correlate with winning outcomes. Through empirical analysis and experimentation, this report presents the efficacy and limitations of the predictive models developed, offering a foundation for further research and exploration in the field of chess analytics and artificial intelligence.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Introduction . . . . .	2
1.2	Aim and Objectives . . . . .	2
1.3	Dataset selected . . . . .	2
1.4	Proposed System . . . . .	3
<b>2</b>	<b>Data preprocessing</b>	<b>4</b>
2.1	Dropping of unnecessary columns . . . . .	4
2.2	One hot encoding . . . . .	4
2.3	Plotting and removing outliers . . . . .	5
<b>3</b>	<b>EDA and visualization</b>	<b>6</b>
3.1	Pie Chart . . . . .	6
3.2	Scatter Chart . . . . .	7
3.3	Box Plot . . . . .	7
3.4	Histograms . . . . .	7
<b>4</b>	<b>Splitting dataset and training models</b>	<b>9</b>
4.1	Splitting dataset . . . . .	9
4.2	Logistic Regression . . . . .	9
4.3	Random Forest Regression . . . . .	10
4.4	Decision Tree Algorithm . . . . .	11
<b>5</b>	<b>Block Diagram</b>	<b>12</b>
<b>6</b>	<b>Frontend / model implementation</b>	<b>13</b>
<b>7</b>	<b>Conclusion</b>	<b>16</b>
7.1	Results . . . . .	16
7.2	Summary . . . . .	16
7.3	Future Scope . . . . .	16
<b>8</b>	<b>References</b>	<b>17</b>

# Chapter 1

## Introduction

### 1.1 Introduction

The dataset provided is a collection of over 20,000 chess games from users on Lichess.org, encompassing various game attributes such as player IDs, moves, outcomes, time controls, ratings, and opening information. This data is structured in a tabular format with each row representing an individual game and columns detailing different game-related features. The objective of this analysis is to leverage this dataset to develop a predictive model using data mining techniques, particularly focusing on classification tasks to predict the outcomes of chess games based on the provided features.

### 1.2 Aim and Objectives

The aim of this project is to build a predictive model that accurately forecasts the outcomes of chess games. This involves employing classification algorithms to categorize each game into one of the possible outcome classes, such as determining the winner or predicting whether a game is rated or not. The specific objectives include understanding and cleaning the dataset to ensure data quality, exploring relationships between variables to uncover patterns that influence game outcomes, and ultimately implementing and evaluating classification algorithms to achieve the highest accuracy in predicting game results.

### 1.3 Dataset selected

The dataset available at the provided link is related to chess games. It likely contains information about various aspects of chess games such as players, moves, outcomes, time controls, ratings, and possibly other relevant features. The dataset could be structured in tabular format with rows representing individual games and columns representing different attributes or features associated with each game. This is a set of just over 20,000 games collected from a selection of users on the site Lichess.org, and how to collect more. I will also upload more games in the future as I collect them. This set contains the:

Game ID;  
Rated (T/F);  
Start Time;

End Time;  
Number of Turns;  
Game Status;  
Winner;  
Time Increment;  
White Player ID;  
White Player Rating;  
Black Player ID;  
Black Player Rating;  
All Moves in Standard Chess Notation;  
Opening Eco (Standardized Code for any given opening, list here);  
Opening Name;  
Opening Ply (Number of moves in the opening phase)

## 1.4 Proposed System

The proposed system will involve a series of steps including data exploration, cleaning, feature engineering, and model development. Initially, we will examine the dataset's structure, handle missing values, and remove duplicates to ensure data integrity. Next, we will conduct correlation analysis and visualize relationships between variables to understand the factors influencing game outcomes. Feature engineering techniques may be applied to derive new meaningful features from the existing ones. Subsequently, various classification algorithms such as logistic regression, decision trees, random forests, and support vector machines will be implemented and evaluated for their predictive performance. The model with the highest accuracy will be selected as the final predictive tool for chess game outcome prediction.



# Chapter 2

## Data preprocessing

### 2.1 Dropping of unnecessary columns

Unnecessary columns are dropped to streamline the dataset and focus only on relevant information, reducing noise and improving model efficiency. This step involves removing columns that do not contribute significantly to the predictive task, enhancing the quality and effectiveness of subsequent analysis and modeling.

```
# Approach 1: Drop missing values
dataset_dropped = dataset_dirty.dropna()
```

```
[ ] print("\nDataset after Dropping Missing Values:")
    print(dataset_dropped)
```

```
Dataset after Dropping Missing Values:
   id  rated  created_at  last_move_at  turns  victory_status \
1  11NXvwaE   True  1.504130e+12  1.504130e+12   16.0         resign
8  dwF3DJHO   True  1.503510e+12  1.503510e+12   66.0         resign
27 srz9QfSN   True  1.502780e+12  1.502780e+12   54.0          mate
33  F1mRjzPr  False  1.502720e+12  1.502720e+12   21.0         resign
46  27zreJYy  False  1.499330e+12  1.499330e+12    3.0         resign
...  ...    ...    ...    ...    ...    ...
20024 ItHcx2zg   True  1.504257e+12  1.504257e+12  118.0          mate
20027 3ium8obe   True  1.504103e+12  1.504104e+12   47.0          mate
20038 xAQi2hl2   True  1.501011e+12  1.501012e+12   36.0         resign
20052 EopEqqAa   True  1.499812e+12  1.499812e+12   37.0         resign
20057 N8G2JHGG   True  1.499643e+12  1.499644e+12   78.0          mate

   winner increment_code  white_id  white_rating \
1    black           5+10      a-00      1322.0
8    black           15+0  ehabfanri      1439.0
27   black           10+10    mannat1      1328.0
33   black           15+3  shivangithegenius  1019.0
46   black           15+0  shivangithegenius   978.0
...  ...    ...    ...    ...
20024 black           10+0      javi_r      1713.0
20027 white           10+0  alexandre789      1842.0
20038 black           15+15    jamboger      1247.0
20052 white           10+10    jamboger      1219.0
20057 black           10+0    jamboger      1235.0

   black_id  black_rating \
```

### 2.2 One hot encoding

One-hot encoding is done to convert categorical variables into a numerical format that can be used as input for machine learning algorithms.

```
[ ] #One-Hot Encoding Categorical Variables
# Assuming 'categorical_column' is the column you want to one-hot encode
dataset_encoded = pd.get_dummies(dataset, columns=['new'], prefix='category')

[ ] #One-Hot Encoding Categorical Variables
# Assuming 'categorical_column' is the column you want to one-hot encode
dataset_encoded = pd.get_dummies(dataset, columns=['turns'], prefix='game_lenght')

print(dataset_encoded[['opening_ply']])
```

	opening_ply
0	5
1	4
2	3
3	3
4	5
...	...
20053	2
20054	2
20055	3
20056	4
20057	3

[20058 rows x 1 columns]

## 2.3 Plotting and removing outliers

Plotting the data allows visualization of any outliers, which can then be identified based on their deviation from the general trend. Subsequently, outliers were removed to improve the robustness and accuracy of the predictive model.

```
[ ] # Replace 'numeric_column' with the actual numerical column you want to analyze
z_scores = zscore(dataset['turns'])
outliers = (np.abs(z_scores) > 2) # Adjust the threshold as needed

# Display identified outliers
print("Identified Outliers:")
print(dataset_cleaned[outliers])
```

Identified Outliers:  
Empty DataFrame  
Columns: [id, rated, created\_at, last\_move\_at, turns, victory\_status, winner, increment\_code, white\_id, white\_rating, black\_id,  
Index: []  
<ipython-input-77-cb7665aae751>:7: UserWarning: Boolean Series key will be reindexed to match DataFrame index.  
print(dataset\_cleaned[outliers])

```
[ ] # Compare summary statistics before and after handling noisy values
print("\nSummary Statistics - Original Dataset:")
print(dataset_dirty['turns'].describe())

print("\nSummary Statistics - Cleaned Dataset:")
print(dataset_cleaned['turns'].describe())
```

Summary Statistics - Original Dataset:

count	20058.000000
mean	0.665816
std	0.037847
min	0.352793
25%	0.657456
50%	0.673491
75%	0.686606
max	0.729206
Name: turns, dtype: float64	

# Chapter 3

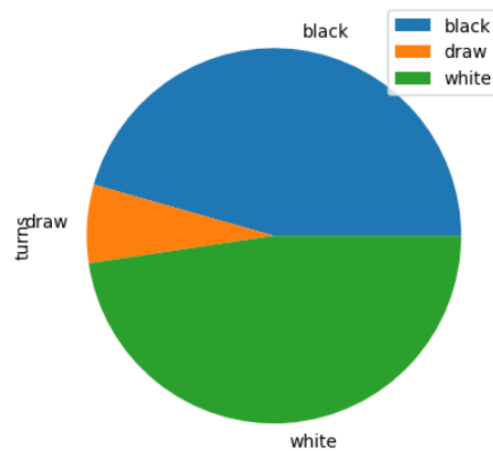
## EDA and visualization

Exploratory Data Analysis (EDA) is an important step in data mining because it helps to uncover insights, patterns, and relationships within the data, allowing for better understanding of the underlying structure and characteristics of the dataset. We have done this crucial step using Heatmap and Histograms.

### 3.1 Pie Chart

```
[ ] df.groupby(['winner']).sum().plot(kind='pie',y='turns')
```

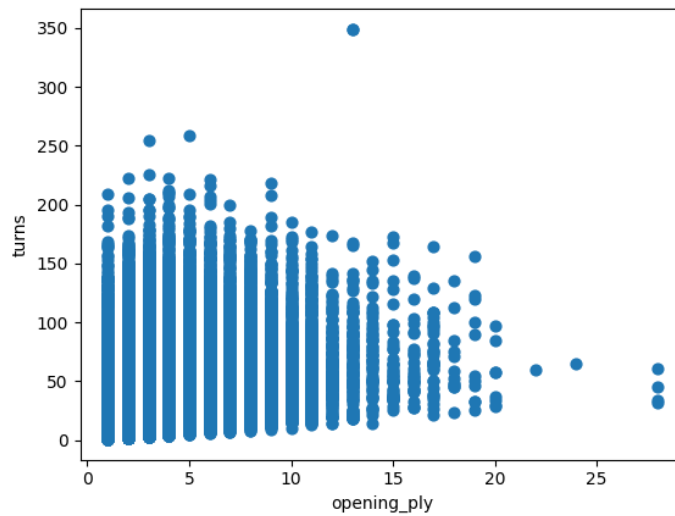
```
<ipython-input-40-e8196f2a3387>:1: FutureWarning: The default value of numeric_only in Dat  
df.groupby(['winner']).sum().plot(kind='pie',y='turns')  
<Axes: ylabel='turns'>
```



Pie Chart

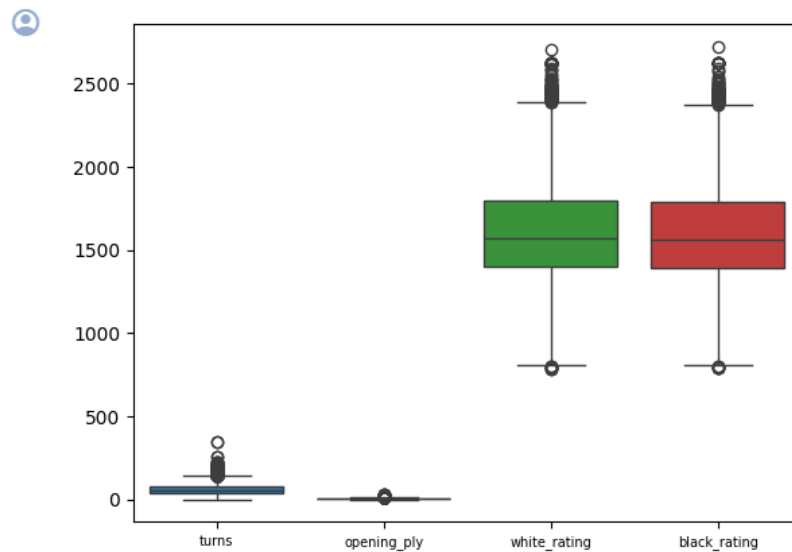
## 3.2 Scatter Chart

```
[ ] df.plot.scatter(x='opening_ply',y='turns',s=40);
```



Scatter Chart

## 3.3 Box Plot



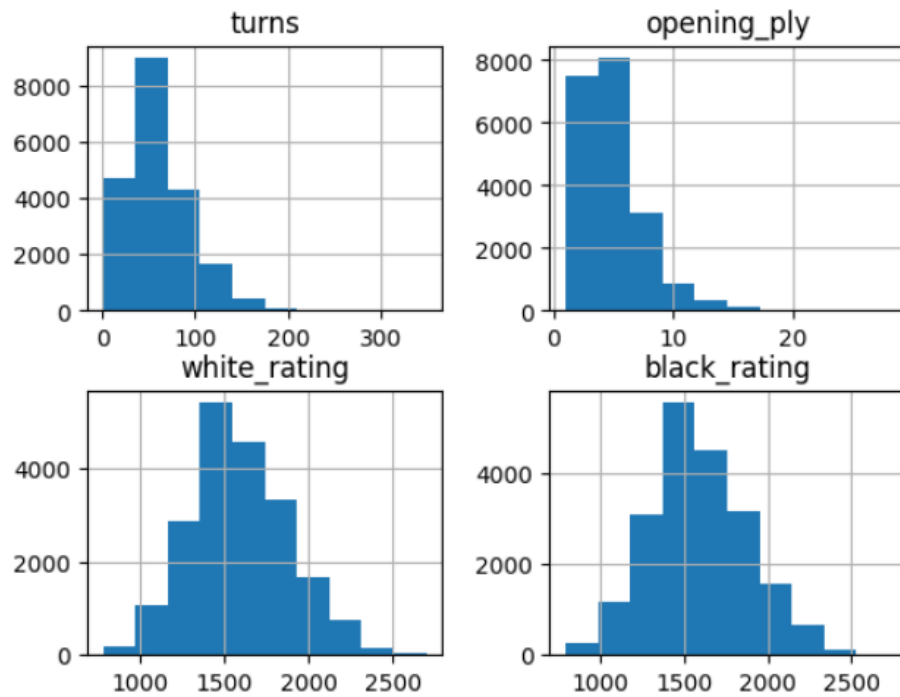
Box Plot

## 3.4 Histograms

Visualizing the occurrence of values in a feature via histograms facilitates understanding the data's central tendency, spread, and shape, crucial for assessing its suitability for modeling and guiding data transformation choices. Here are our visualization results:

```
[ ] # Subsetting the DataFrame to include only the specified numerical columns
df_subset = df[numerical_columns]
df_subset.hist()
```

```
array([[<Axes: title={'center': 'turns'}>,
        <Axes: title={'center': 'opening_ply'}>],
       [<Axes: title={'center': 'white_rating'}>,
        <Axes: title={'center': 'black_rating'}>]], dtype=object)
```



# Chapter 4

## Splitting dataset and training models

### 4.1 Splitting dataset

The data is split into training and testing sets, with the training dataset and the testing dataset.

```
1 d4 Nc6 e4 e5 f4 f6 dxe5 fxe5 fxe5 Nxe5 Qd4 Nc6... B00
2 e4 e5 d3 d6 Be3 c6 Be2 b5 Nd2 a5 a4 c5 axb5 Nc... C20
3 d4 d5 Nf3 Bf5 Nc3 Nf6 Bf4 Ng4 e3 Nc6 Be2 Qd7 O... D02
4 e4 e5 Nf3 d6 d4 Nc6 d5 Nb4 a3 Na6 Nc3 Be7 b4 N... C41
```

```
opening_name opening_ply
0 Slav Defense: Exchange Variation 5
1 Nimzowitsch Defense: Kennedy Variation 4
2 King's Pawn Game: Leonardis Variation 3
3 Queen's Pawn Game: Zukertort Variation 3
4 Philidor Defense 5
```

```
[ ] # Encode categorical variables
label_encoder = LabelEncoder()
data['victory_status'] = label_encoder.fit_transform(data['victory_status'])
data['increment_code'] = label_encoder.fit_transform(data['increment_code'])
data['opening_eco'] = label_encoder.fit_transform(data['opening_eco'])
data['opening_name'] = label_encoder.fit_transform(data['opening_name'])

# Define features (X) and target variable (y)
X = data.drop(['id', 'rated', 'winner', 'white_id', 'black_id', 'moves'], axis=1)
y = label_encoder.fit_transform(data['winner'])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

### 4.2 Logistic Regression

Logistic regression is a classification algorithm used to predict binary outcomes (yes/no, win/lose) by estimating probabilities. It uses a logistic function to transform inputs into probabilities between 0 and 1, making it suitable for binary classification tasks such as predicting match outcomes in sports analytics.

```

from sklearn.linear_model import LogisticRegression

# Initialize and train the Logistic Regression model
logreg_model = LogisticRegression(max_iter=1000)
logreg_model.fit(X_train, y_train)

# Make predictions and evaluate the model
logreg_preds = logreg_model.predict(X_test)
logreg_accuracy = accuracy_score(y_test, logreg_preds)
print("Logistic Regression Accuracy:", logreg_accuracy)
print("Logistic Regression Classification Report:")
print(classification_report(y_test, logreg_preds))

```

Logistic Regression Accuracy: 0.4995014955134596

Logistic Regression Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1816
1	0.00	0.00	0.00	192
2	0.50	1.00	0.67	2004
accuracy			0.50	4012
macro avg	0.17	0.33	0.22	4012
weighted avg	0.25	0.50	0.33	4012

The logistic regression model achieved an accuracy of approximately 50 percent, indicating that it performs no better than random guessing for this classification task.

The precision, recall, and F1-score metrics provide further insights into the model's performance:

Precision measures the accuracy of the positive predictions. For class 0 and class 1, the precision is 0 percent, indicating that the model did not correctly predict any instances for these classes. However, for class 2, the precision is 50 percent, suggesting that half of the predicted instances for this class were correct.

### 4.3 Random Forest Regression

The Random Forest algorithm is an ensemble learning method that constructs a multitude of decision trees during training. Each tree is built on a random subset of features and data points, and predictions are made by aggregating the results from individual trees (bagging). Random Forests are robust against overfitting, handle high-dimensional data well, and provide feature importance measures.

```

from sklearn.ensemble import RandomForestClassifier

# Initialize and train the Random Forest Classifier model
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

# Make predictions and evaluate the model
rf_preds = rf_model.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_preds)
print("Random Forest Classifier Accuracy:", rf_accuracy)
print("Random Forest Classifier Classification Report:")
print(classification_report(y_test, rf_preds))

```

Random Forest Classifier Accuracy: 0.6914257228315055

Random Forest Classifier Classification Report:

	precision	recall	f1-score	support
0	0.67	0.63	0.65	1816
1	1.00	0.94	0.97	192
2	0.68	0.72	0.70	2004
accuracy			0.69	4012
macro avg	0.78	0.77	0.77	4012
weighted avg	0.69	0.69	0.69	4012

## 4.4 Decision Tree Algorithm

A decision tree algorithm is a popular method in machine learning used for both classification and regression tasks. Its structure resembles a tree, where each internal node represents a decision based on a feature, and each leaf node represents the predicted outcome. The algorithm works by recursively splitting the data based on the most informative features, chosen using criteria like Gini impurity or mean squared error. This process continues until a stopping criterion is met, such as reaching a maximum depth or having too few instances in a node. Predictions are made by traversing the tree from the root to a leaf node based on the input features of a new data point, and the output is determined by the majority class (for classification) or average value (for regression) in that leaf node. Decision trees are valued for their simplicity, interpretability, and ability to handle both numerical and categorical data, although they can be prone to overfitting complex data.

```

from sklearn.tree import DecisionTreeClassifier

# Initialize and train the Decision Tree Classifier model
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)

# Make predictions and evaluate the model
dt_preds = dt_model.predict(X_test)
dt_accuracy = accuracy_score(y_test, dt_preds)
print("Decision Tree Classifier Accuracy:", dt_accuracy)
print("Decision Tree Classifier Classification Report:")
print(classification_report(y_test, dt_preds))

```

```

Decision Tree Classifier Accuracy: 0.6358424725822532
Decision Tree Classifier Classification Report:

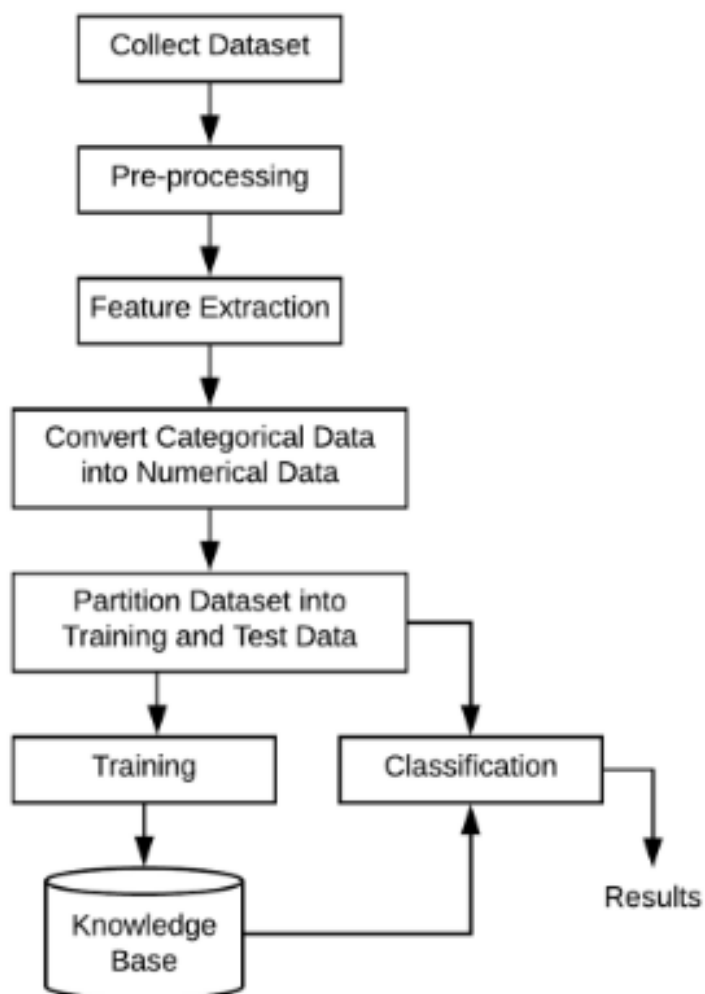
```

	precision	recall	f1-score	support
0	0.60	0.60	0.60	1816
1	0.97	0.94	0.96	192
2	0.64	0.64	0.64	2004
accuracy			0.64	4012
macro avg	0.74	0.73	0.73	4012
weighted avg	0.64	0.64	0.64	4012



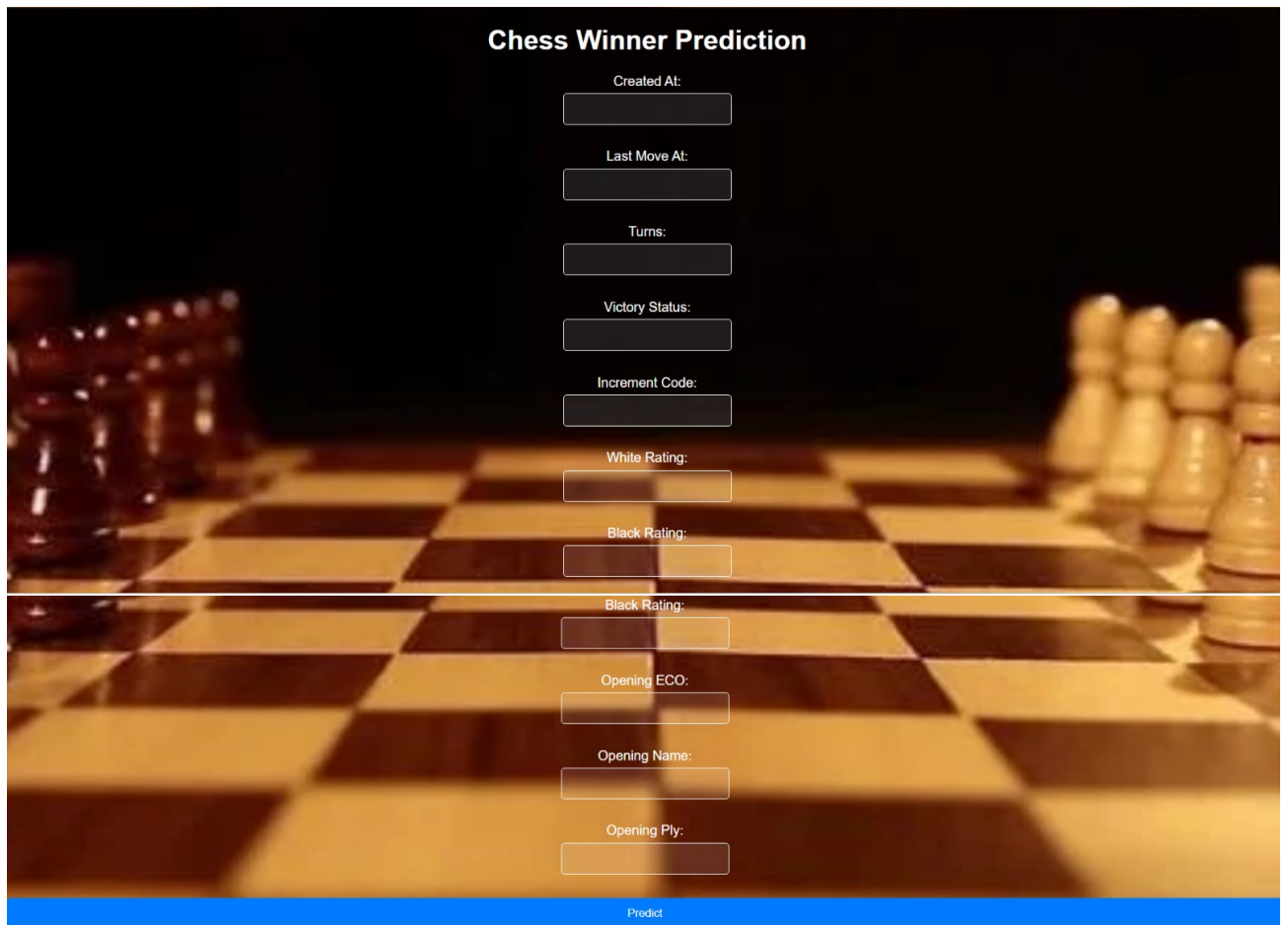
# Chapter 5

## Block Diagram



# Chapter 6

## Frontend / model implementation



**Chess Winner Prediction**

Created At:

Last Move At:

Turns:

Victory Status:

Increment Code:

White Rating:

Black Rating:

Black Rating:

Opening ECO:

Opening Name:

Opening Ply:

### Chess Winner Prediction

Created At:	1.50E+12
Last Move At:	1.5E+12
Turns:	16
Victory Status:	mate
Increment Code:	10+0
White Rating:	1500
Black Rating:	1106

## Prediction Result

Predicted Winner: white



White wins!

[Go back](#)

## Prediction Result

Predicted Winner: black



Black wins!

[Go back](#)

# Chapter 7

## Conclusion

### 7.1 Results

Based on these metrics, the XGBoost appears to be better suited for this particular problem: The project involved developing a web application using Flask and integrating an XGBoost model to predict chess game winners. Data preprocessing, model training, and serialization were done using Pandas, Scikit-Learn, and Joblib. The Flask app had routes for home and prediction, with a user-friendly HTML/CSS interface. Key learnings included ML model integration, frontend design, and deployment.

### 7.2 Summary

The project involved building a web application using Flask to predict winners in chess games. The application leveraged an XGBoost model trained on historical chess game data. Key technologies included Pandas for data manipulation, Scikit-Learn for preprocessing and model training, and Joblib for model serialization. The frontend was developed using HTML and CSS, providing intuitive interfaces for users to input game details and receive predictions. The focus was on integrating the machine learning model seamlessly with the user interface, ensuring accuracy and efficiency in predicting game outcomes. Future iterations could enhance the user experience with improved UI design, input validation mechanisms, error handling, and potentially user authentication features for personalized interactions.

### 7.3 Future Scope

In the future, this chess prediction web application can be expanded and improved in several ways. One potential area for enhancement is the user interface, where more interactive and visually appealing elements can be added to provide a better user experience. This includes features such as real-time updates, visualizations of game statistics, and personalized user profiles. Additionally, implementing input validation mechanisms can help ensure that users provide accurate and valid data, improving the reliability of the predictions.

[hyperref](#)

# Chapter 8

## References

1. **Data Preprocessing in Machine learning :** <https://www.javatpoint.com/data-preprocessing-machine-learning>
2. **Data Visualization:** <https://www.domo.com/charts/bar-charts>
3. **Linear Regression :** <https://www.javatpoint.com/linear-regression-in-machine-learning>
4. **Random Forest Algorithm :** <https://www.javatpoint.com/machine-learning-random-forest-algorithm>