**Project Description**

1. **Aim of the Project:**

The objective of this project is to develop a machine learning model to predict the winner of an IPL match based on various match-related features such as teams playing, toss winner, toss decision, venue, and other relevant factors.

By training and evaluating different machine learning models—including Logistic Regression, SVM, KNN, Decision Trees, Random Forest, and XGBoost—the project aims to identify the most accurate model for predicting match outcomes. Additionally, hyperparameter tuning will be performed to optimize the best model.

The final trained model will be saved in a .pkl file, and the entire process will be documented with a report and PowerPoint presentation summarizing the findings.

1. **Business Problem or Problem Statement:**

The **Indian Premier League (IPL)** is one of the most popular and commercially successful cricket leagues globally, attracting millions of viewers and generating significant revenue through sponsorships, advertising, and betting markets. However, predicting match outcomes remains a challenge due to multiple influencing factors such as team composition, toss results, match venue, and past performances.

**Problem:**  
Stakeholders—including teams, analysts, fantasy league players, and sports bettors—lack a **data-driven approach** to accurately predict match winners. Traditional methods rely heavily on expert opinions and subjective analysis, which can be biased or inconsistent.

1. **Project Description:**

**Overview:**

The **Indian Premier League (IPL)** is one of the most competitive cricket leagues, where predicting match winners is a complex task due to multiple influencing factors such as team composition, toss results, venue conditions, and past performances. This project aims to develop a **machine learning model** that accurately predicts the winner of an IPL match based on historical match data.

**Objectives:**

* Load and preprocess IPL match data, ensuring data quality and consistency.
* Perform **Exploratory Data Analysis (EDA)** to identify key patterns and insights.
* Encode categorical variables and prepare datasets for machine learning models.
* Train and compare multiple machine learning models, including **Logistic Regression, SVM, Decision Trees, Random Forest, and XGBoost**.
* Optimize the best-performing model using **hyperparameter tuning** to improve prediction accuracy.
* Evaluate models based on metrics such as **accuracy, precision, recall, and F1-score**.
* Save the final trained model in a **.pkl file** for future use.
* Document the entire process and present the findings through a **detailed report and PowerPoint presentation**.

**Methodology:**

1. **Data Collection & Understanding** – Load and explore IPL match data.
2. **Data Cleaning & Preprocessing** – Handle missing values, encode categorical features, and normalize numerical data.
3. **Exploratory Data Analysis (EDA)** – Identify trends in team performance, toss influence, and match outcomes using visualizations.
4. **Feature Engineering** – Select and transform relevant features for model training.
5. **Model Training & Evaluation** – Train multiple ML models and compare their performance.
6. **Hyperparameter Tuning** – Optimize the best model for improved accuracy.
7. **Model Deployment & Documentation** – Save the final model and document the process for future use.

**4 . Functionalities:**

The project consists of multiple functionalities that enable data processing, model training, evaluation, and prediction of IPL match outcomes. Below are the key functionalities:

**1. Data Loading & Preprocessing**

* Load IPL dataset into a **pandas DataFrame**.
* Handle **missing values** and clean inconsistencies in data.
* Convert categorical features like **team names, toss decisions, and venues** into numerical format using **Label Encoding / One-Hot Encoding**.
* Normalize or scale numerical features if required.

**2. Exploratory Data Analysis (EDA)**

* Perform **statistical analysis** to understand trends in match results.
* Visualize factors affecting match outcomes using **bar charts, histograms, and heatmaps**.
* Analyze the impact of **toss winners, toss decisions, and venue advantage** on match results.
* Compute feature correlations to identify **important factors** influencing match outcomes.

**3. Feature Engineering & Data Splitting**

* Select relevant **features (X)** and define the **target variable (y)** (winner).
* Split the dataset into **training (80%) and testing (20%)** sets for model evaluation.
* Apply **feature scaling** if necessary.

**4. Model Training & Comparison**

* Train multiple **machine learning models**:
  + Logistic Regression
  + Support Vector Machine (SVM)
  + Decision Trees
  + Random Forest
  + XGBoost
* Compare model performance using **accuracy, precision, recall, and F1-score**.

**5. Hyperparameter Tuning**

* Optimize the best-performing model using **Grid Search or Random Search**.
* Fine-tune hyperparameters to improve **model accuracy**.

**6. Model Evaluation & Selection**

* Generate a **confusion matrix** to analyze prediction errors.
* Compare performance metrics and select the **best model** for IPL winner prediction.

**7. Prediction Functionality**

* Accept **new match details (teams, toss winner, venue, etc.)** as input.
* Process input data to match the trained model’s format.
* Use the trained model to **predict the match winner**.
* Display the predicted team along with model confidence score.

1. **Code Implementation:**

To implement the project, we utilize basic Python programming concepts to create a modular and maintainable codebase. We leverage key algorithms and data structures to efficiently manage data processing tasks. The code is organized into modules to ensure modularity and readability, with extensive documentation provided for clarity and future development.

**Description:**

In this project, we implement various modules using basic Python programming concepts. Each module is designed to handle specific functionalities of the school management system.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split, GridSearchCV

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

import pickle

# 1. Load Dataset

df = pd.read\_csv("ipl\_matches.csv") # Replace with actual dataset path

print("Dataset Loaded Successfully")

print(df.head()) # Display first few rows

# 2. Data Preprocessing

# Handling missing values by forward filling

df.fillna(method='ffill', inplace=True)

# Encoding categorical variables using Label Encoding

le = LabelEncoder()

df['team1'] = le.fit\_transform(df['team1'])

df['team2'] = le.transform(df['team2'])

df['toss\_winner'] = le.transform(df['toss\_winner'])

df['winner'] = le.transform(df['winner'])

df['toss\_decision'] = df['toss\_decision'].map({'bat': 0, 'field': 1})

# Selecting relevant features for prediction

features = ['team1', 'team2', 'toss\_winner', 'toss\_decision']

X = df[features] # Feature matrix

y = df['winner'] # Target variable

# Splitting dataset into training and testing sets (80% training, 20% testing)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardizing data using StandardScaler

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# 3. Model Training

# Defining multiple models for comparison

models = {

'Logistic Regression': LogisticRegression(),

'SVM': SVC(),

'KNN': KNeighborsClassifier(),

'Decision Tree': DecisionTreeClassifier(),

'Random Forest': RandomForestClassifier(),

'XGBoost': XGBClassifier()

}

# Training each model and evaluating accuracy

for name, model in models.items():

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

acc = accuracy\_score(y\_test, y\_pred)

print(f"{name} Accuracy: {acc:.4f}")

# 4. Model Selection and Hyperparameter Tuning

# Choosing Random Forest as the best model and performing hyperparameter tuning

best\_model = RandomForestClassifier()

param\_grid = {

'n\_estimators': [50, 100, 200],

'max\_depth': [None, 10, 20],

'min\_samples\_split': [2, 5, 10]

}

# Using GridSearchCV for hyperparameter tuning

grid\_search = GridSearchCV(best\_model, param\_grid, cv=5, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

print("Best Parameters:", grid\_search.best\_params\_)

# Setting the best model obtained from GridSearchCV

best\_model = grid\_search.best\_estimator\_

# 5. Model Evaluation

# Predicting on test data and evaluating performance

y\_pred = best\_model.predict(X\_test)

print("Final Model Accuracy:", accuracy\_score(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

# Saving the trained model using pickle

with open("ipl\_winner\_model.pkl", "wb") as f:

pickle.dump(best\_model, f)

print("Model Saved Successfully")

# 6. Prediction Function

def predict\_match(team1, team2, toss\_winner, toss\_decision):

"""

Function to predict the winner of an IPL match.

:param team1: Name of first team

:param team2: Name of second team

:param toss\_winner: Name of team that won the toss

:param toss\_decision: Toss decision ('bat' or 'field')

:return: Predicted winner team name

"""

team1 = le.transform([team1])[0]

team2 = le.transform([team2])[0]

toss\_winner = le.transform([toss\_winner])[0]

toss\_decision = 0 if toss\_decision.lower() == 'bat' else 1

input\_data = np.array([[team1, team2, toss\_winner, toss\_decision]])

input\_data = scaler.transform(input\_data)

prediction = best\_model.predict(input\_data)

return le.inverse\_transform(prediction)[0]

# Example Usage

print("Predicted Winner:", predict\_match("Mumbai Indians", "Chennai Super Kings", "Mumbai Indians", "bat"))

**7 . Results and Outcomes:**

Comparing the models performance on Comparision with each other's accuracy score XG Boost is having the highest score.

XG Boost has got the accuracy of 94%.

Random forest Classifier has got accuracy of 83%.

Logistic Regressor has got accuracy of 56.58%

SVM has achieved training accuracy of 62.07% and test accuracy of 65.78%.

DT(Decision Tree Classifier) has achieved accuracy of 41%.

**8 . Conclusion:**

XGBoost is the most effective model in this comparison, achieving the highest accuracy of **94%**, making it the best choice for the given dataset. Other models like Random Forest performed reasonably well, while Decision Tree and Logistic Regression showed lower accuracy, indicating they may not be suitable for this task.

* The team with the highest number of wins is **Mumbai Indians (MI)**, followed by **Chennai Super Kings (CSK)**.
* **Chris Gayle** has won the most "Player of the Match" awards.
* His best year was **2011**, where he secured **6 awards**.
* Matches are played across many venues, but **Eden Gardens, Wankhede Stadium, and M. Chinnaswamy Stadium** host the most games.
* Most matches are won by **wickets (chasing teams winning)** rather than runs.