Moneyball Part Deux Part II

Predicting the Rockies 2024 remaining season using ML Models.

9/11/2024

Team

Liz Cooney

Howard Dixon

Cassy Miller

Sandokan Stage

1

Introduction:

1. Team
2. Executive Summary
3. Data Overview + Approach

2

Models: Cassy

1. Linear
2. Logistic

3

Models: Liz

1. Decision Tree
2. Random Forest

4

Models: Howard

1. KNN

5

Models: Sando

1. SVM

6

Results:

1. Conclusion
2. Questions

AGENDA

Executive Summary

Objective

In this project, we set out to predict the remaining games of the 2024 season for the Colorado Rockies by harnessing the power of machine learning. Drawing from an extensive dataset of game statistics spanning from 2013 to 2024, we analyzed a variety of performance indicators, including win-loss records, hitting statistics, home runs, strikeouts, and opponent information. By leveraging these data points, our goal was to uncover patterns and trends that could help forecast the team's future performance. This endeavor is not just about numbers but about capturing the essence of the game and the factors that could influence the Rockies' journey through the rest of the season.

Our approach was comprehensive and multi-faceted, employing several machine learning models, including Decision Trees, Random Forests, Linear and Logistic Regression, Support Vector Machines (SVM), and K-Nearest Neighbors (KNN). Each model was carefully selected for its unique strengths in handling the complexity and variability inherent in baseball data. By training these models on historical data, we sought to identify the most influential factors impacting game outcomes and to create a predictive framework that accurately reflects the team's performance potential. Our commitment to rigorous analysis and model validation ensures that our predictions are not only statistically sound but also meaningful for strategic decision-making.

The results of our analysis offer a nuanced view of what may lie ahead for the Colorado Rockies. By understanding the subtle dynamics at play—whether it's the impact of home and away games, day versus night performances, or the influence of specific opponents—we aim to provide actionable insights that can guide coaching decisions, player development, and overall team strategy. Our findings are more than just forecasts; they are a testament to the potential of data-driven decision-making in sports and reflect our deep dedication to understanding and predicting the game’s ever-evolving nature.

Executive Summary

Data Overview and Approach

- Tree-Based Models:
  - **Decision Trees:** Used to explore the branching possibilities of game outcomes based on specific conditions (e.g., opponent strength, game location). This model provides a clear, visual representation of how different factors contribute to winning or losing.
  - **Random Forests:** An ensemble method that aggregates multiple decision trees to improve prediction accuracy. This model is particularly useful for capturing more nuanced relationships between variables and generalizing well across different scenarios.
- Regression Models:
  - **Linear Regression:** Applied to identify linear relationships between game metrics (like hits or home runs) and outcomes. This model helped establish a baseline understanding of how individual variables impact performance.
  - **Logistic Regression:** Used to model the probability of categorical outcomes (win or loss). This approach provides a probabilistic perspective, estimating the likelihood of various game results based on historical data.
- Advanced and Proximity-Based Models:
  - **Support Vector Machines (SVM):** Employed to handle more complex, non-linear relationships by finding the optimal boundary that separates different classes (win or loss). SVM is particularly effective in distinguishing outcomes when data points are not linearly separable.
  - **K-Nearest Neighbors (KNN):** Used for its intuitive approach to predicting outcomes based on the similarity of past games. KNN relies on proximity, comparing current games with similar historical games to make predictions.

Process

The team underwent a large data gathering expedition in order to accumulate enough data to preform various ML models.

The data takes basic game stats from (almost) every Colorado Rockies game since 2013, and combines it with pybaseball (a python library used for scraping baseball stats) to retrieve specific game statistics including obp, hits, strikeouts etc.

The goal was to have a unified and clean dataframe for the team to use on their models.

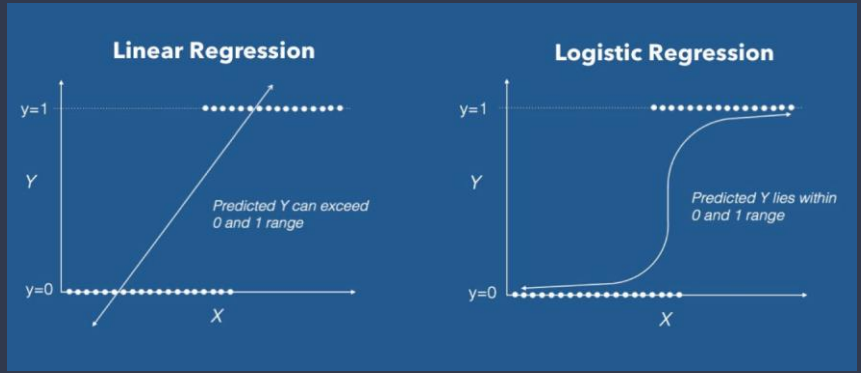
Thank you pybaseball

Linear & Logistic Regression

Description: Models the relationship between input features and a continuous output.

Example: Predicts total runs based on an array of team stats such as obp, hits, and home runs.

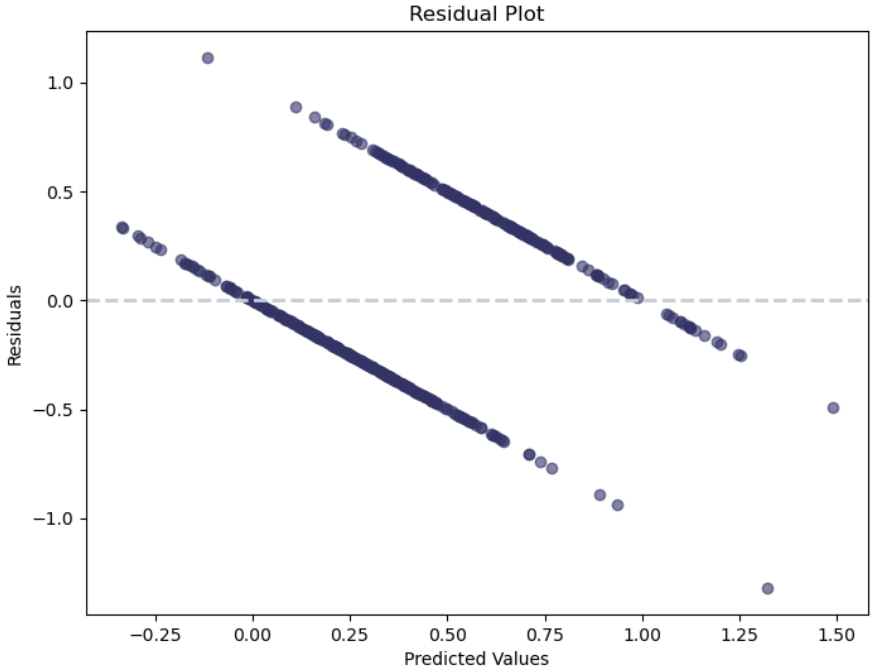
Key Takeaways: Simple, interpretable; good for understanding linear relationships.



Linear Regression

Using the model to predict wins

- Processed data by converting categorical and continuous data.
  - Encoding categorical and converting binary data.
  - Standard Scaler to normalize continuous metrics.



R^2 Score: 0.395 | MSE Score: 0.147

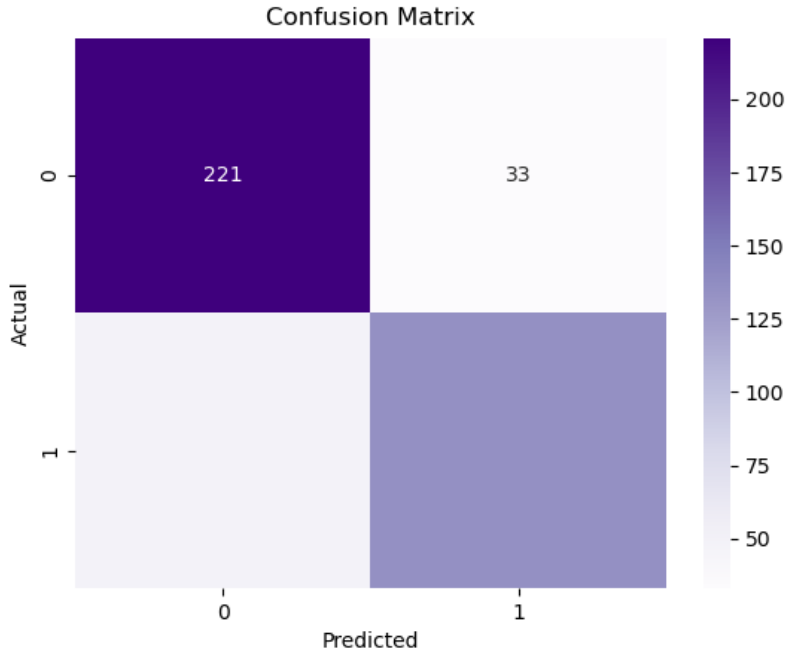
The MSE means our model's predictions are close to the actual values but only 36% of variability is explained by the model.

Pattern shows linear model doesn't work well for the data.

Logistic Regression

Using the model to predict wins

- Processed data by converting categorical and continuous data.
  - Encoding categorical and converting binary data.
  - Standard Scaler to normalize continuous metrics.



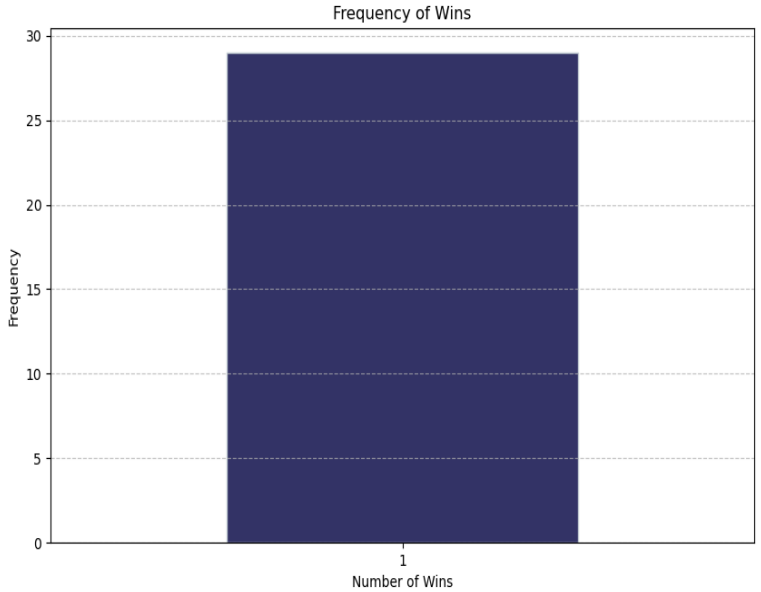
Train Score: 0.791 | Test Score: 0.812

The Log model did perform better than the linear model, most likely do to added complexity of the model. It correctly predicted 221 losses and 135 wins, only mis predicting 49 wins and 33 losses.

Using the Logistic model to actually predict games.

For the remaining 29 games in the season, the logistic model was used to predict W/L based on COL season averages and the Opponent's season averages.

There is one inherent flaw with this method. It relies on much fewer metrics to predict (i.e. D/N and Opponent).



Train Score: 0.248 | Test Score: 0.1803

The Log model did not do well predicting based on the actual unplayed games. As you can see, it predicted all games to be a win, and based on the Rockies track record, this is objectively unrealistic.

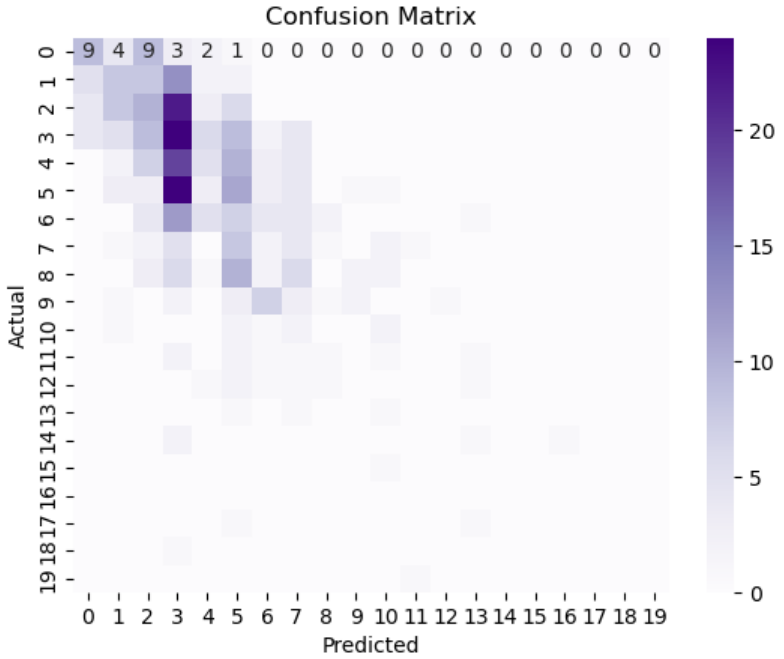
Wrong.

Gm#	September		Tm		Opp	W/L	R	RA
138	Sunday, Sep 1	boxscore	COL		BAL	L	1	6
139	Tuesday, Sep 3	boxscore	COL	@	ATL	L	0	3
140	Wednesday, Sep 4	boxscore	COL	@	ATL	L	2	5
141	Thursday, Sep 5	boxscore	COL	@	ATL	W	3	1
142	Friday, Sep 6	boxscore	COL	@	MIL	W	3	2
143	Saturday, Sep 7	boxscore	COL	@	MIL	L	2	5

Logistic Regression

Using the model to predict **RUNS**

- Processed data by converting categorical and continuous data.
  - Encoding categorical and converting binary data.
  - Standard Scaler to normalize continuous metrics.



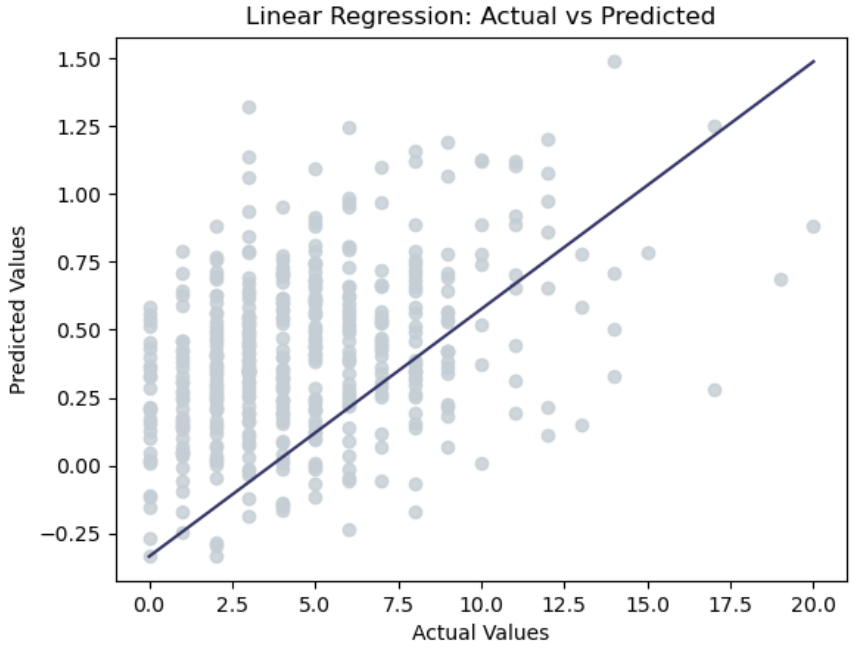
Train Score: 0.248 | Test Score: 0.180

Well. I guess not.

Linear Regression

Using the model to predict **RUNS**

- Processed data by converting categorical and continuous data.
  - Encoding categorical and converting binary data.
  - Standard Scaler to normalize continuous metrics.



R^2 Score: 0.469 | MSE Score: 6.070

Nope.

Conclusion:

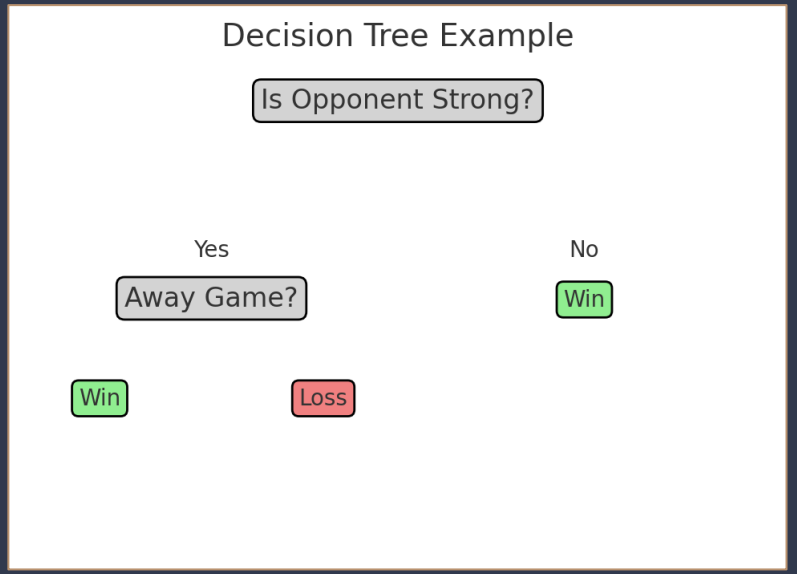
Linear and Logistic models either didn't have the complexity required to accurately predict wins and losses or runs,
OR
We need to shift our approach away from the team as a whole and focus more on specific player data.

Decision Trees & Random Forest

DT: Description: Splits data into branches based on feature values to predict outcomes.
RF: Description: Combines multiple Decision Trees to improve prediction accuracy.

DT: Example: Predicts win/loss based on game location and opponent strength.
RF: Example: Predicts overall win-loss record by analyzing various game conditions.

DT: Key Takeaways: Easy to interpret, prone to overfitting; useful for clear, visual decisions.
RF: Key Takeaways: More accurate and robust than a single tree; handles complex data well.

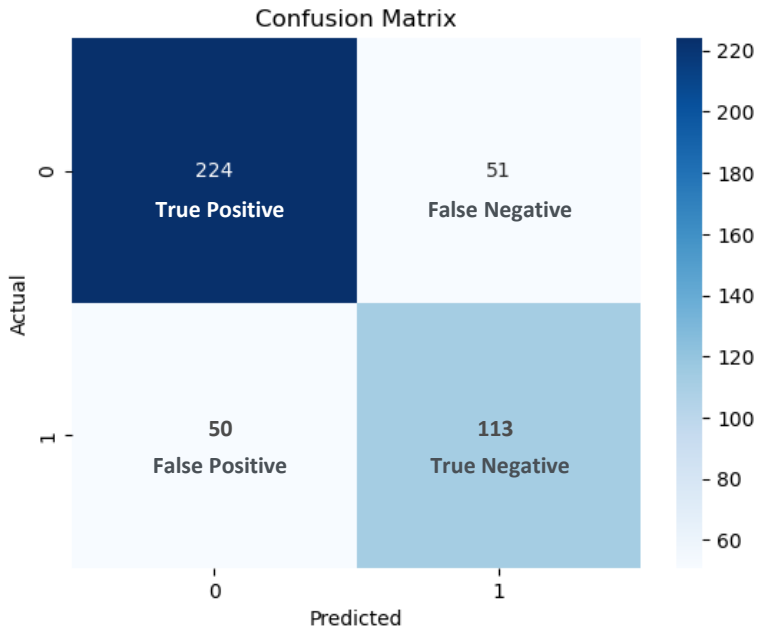


Decision Tree

Trained Data Accuracy: 0.999
Test Data Accuracy: 0.753

- 1751 rows of data
- Target Value – Win/Loss
- 18 Features –
  - team statistical data,
  - opponent statistical data,
  - day/night games,
  - Home vs away

```
# Best Parameters: {'criterion': 'gini',
'max_depth': None, 'min_samples_leaf': 6,
'min_samples_split': 90}
```



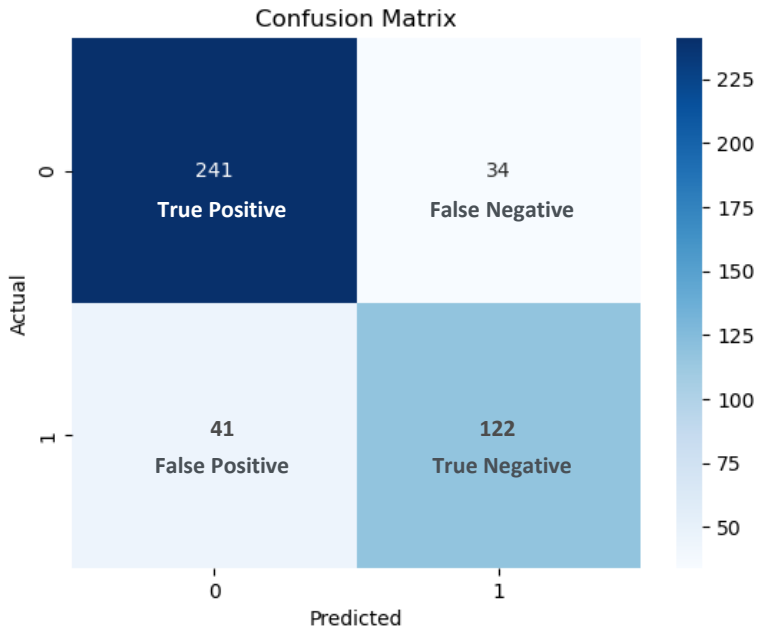
Classification Report:	precision	recall	f1-score	support
0	0.81	0.81	0.81	275
1	0.69	0.69	0.69	163
accuracy			0.77	438
macro avg	0.75	0.75	0.75	438
weighted avg	0.77	0.77	0.77	438

Random Forest

Trained Data Accuracy: 1.000
Test Data Accuracy: 0.820

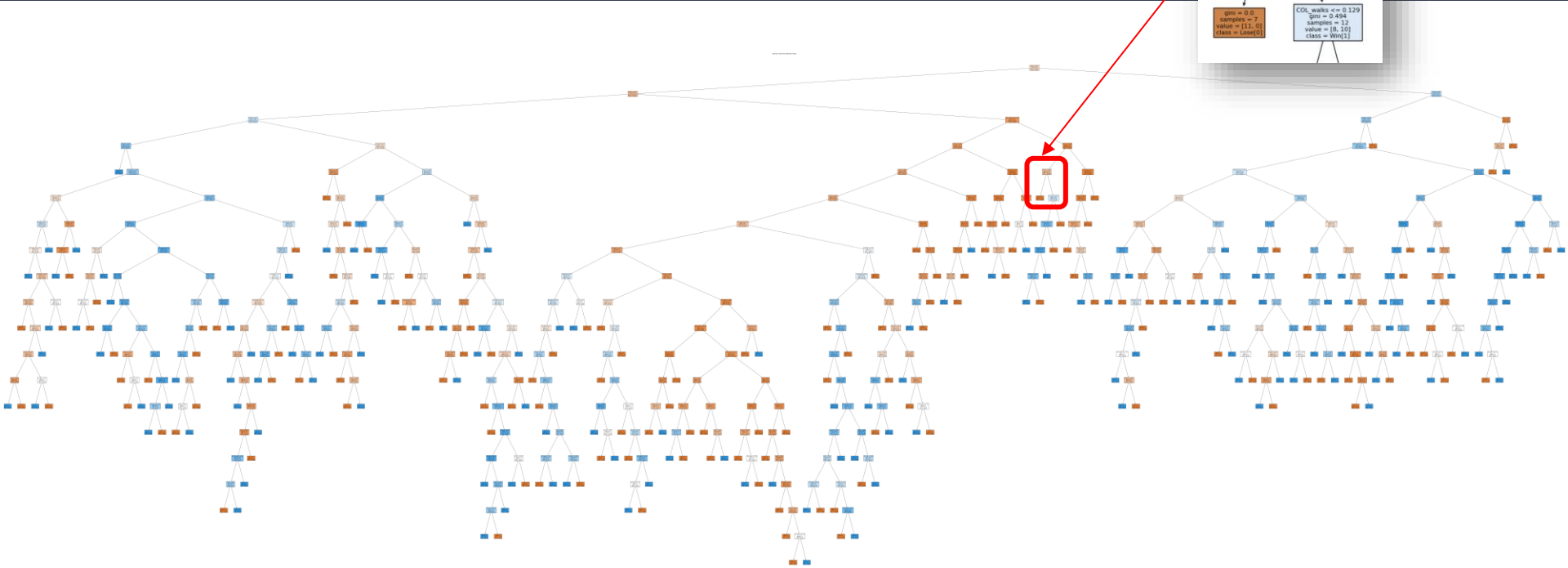
- 1751 rows of data
- Target Value – Win/Loss
- 18 Features –
  - team statistical data,
  - opponent statistical data,
  - day/night games,
  - Home vs away

```
(n_estimators=128, criterion="gini", max_depth=None,
min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0, max_features="auto",
random_state=0)
```

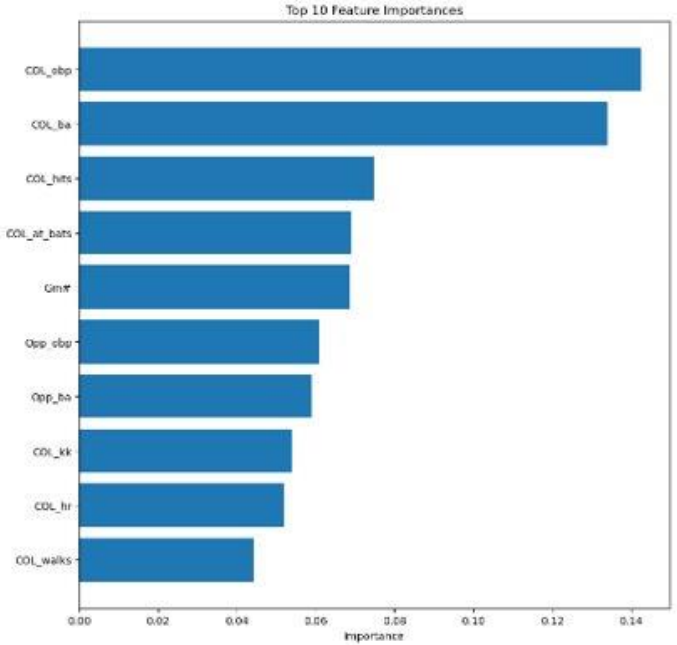


Classification Report:	precision	recall	f1-score	support
0	0.84	0.88	0.86	275
1	0.78	0.72	0.75	163
accuracy			0.82	438
macro avg	0.81	0.80	0.80	438
weighted avg	0.82	0.82	0.82	438

Random Forest - Single Tree



Which features have the greatest impact ?



Colorado Rockies' performance, according the RandomForest model, is most greatly impacted by:

1. The Rockies' on base percentage at each game
2. The Rockies' batting average at each game

Other factors, almost 50% as impactful:

3. The Rockies' total hits per game
4. The Rockies' total at bats per game

Overall:
Rockies' statistics are more important in determining win/loss, over their opponents statistics or which opponent they play against.

Conclusion:

Rockies will lose most games

The team is only predicted to win against the Milwaukee Brewers, Based upon this season's cumulative statistics. The random forest model has only been correct 70**% of the time** for the past 10 games.

Model test score received 82%, which means it can only be wrong between 4-5 times. This model needs more features to better predict the future outcomes, or may not be the right model type.

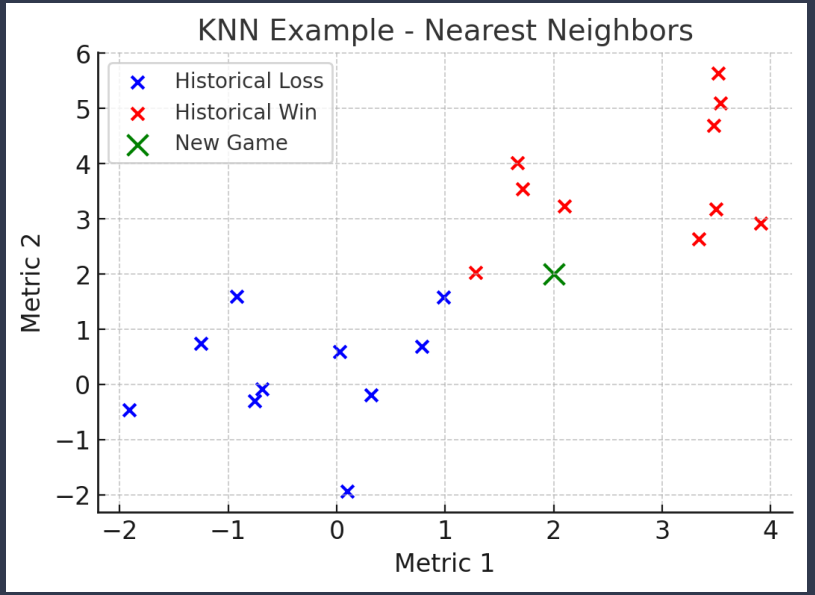
	Formatted_Date	Opp	Win/Loss	
0	2024-08-31	BAL	loss	✖
1	2024-09-01	BAL	loss	
2	2024-09-03	ATL	loss	
3	2024-09-04	ATL	loss	
4	2024-09-05	ATL	loss	✖
5	2024-09-06	MIL	win	
6	2024-09-07	MIL	win	✖
7	2024-09-08	MIL	win	
8	2024-09-10	DET	loss	
9	2024-09-11	DET	loss	
10	2024-09-12	DET	loss	X
11	2024-09-13	CHC	loss	
12	2024-09-14	CHC	loss	
13	2024-09-15	CHC	loss	
14	2024-09-16	ARI	loss	
15	2024-09-17	ARI	loss	
16	2024-09-18	ARI	loss	
17	2024-09-20	LAD	loss	
18	2024-09-21	LAD	loss	
19	2024-09-22	LAD	loss	
20	2024-09-24	STL	loss	
21	2024-09-25	STL	loss	
22	2024-09-26	STL	loss	
23	2024-09-27	LAD	loss	
24	2024-09-28	LAD	loss	
25	2024-09-29	LAD	loss	

KNN (K-Nearest Neighbors)

Description: Predicts outcomes based on the closest historical data points.

Example: Predicts game outcome by comparing it to similar past games.

Key Takeaways: Simple, intuitive; best for localized patterns, depends on ‘k’ choice.



KNN

K-Nearest Neighbors (KNN) Algorithm

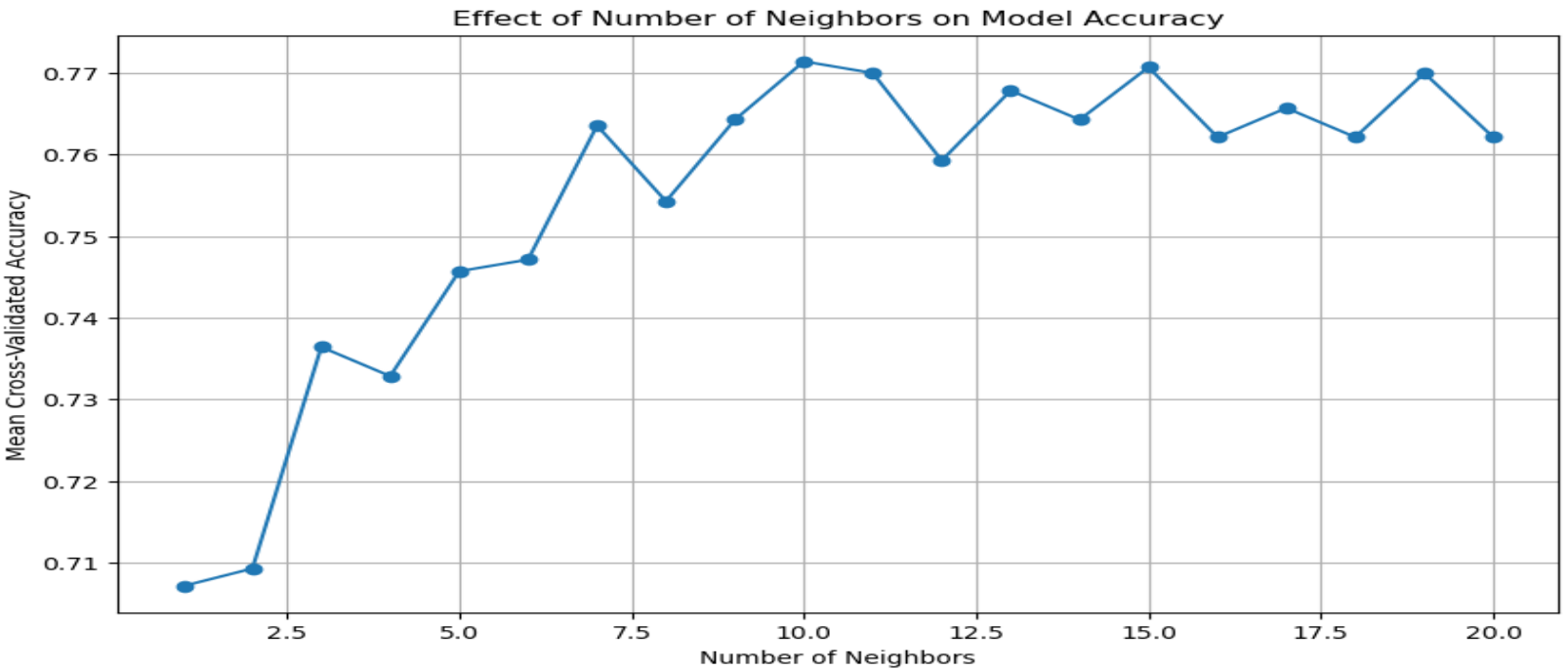
Overview:

- **Type:** Supervised Learning
- **Applications:** Classification, Regression

How It Works:

- **Choose k:** Number of nearest neighbors to consider.
- **Calculate Distance:** Compute distance between data points (e.g., Euclidean, Manhattan).
- **Find Neighbors:** Identify k closest data points to the new data point.
- **Classify/Predict:**
- **Classification:** Assign the most common class label among the k neighbors.
- **Regression:** Calculate the average of k neighbors' values.

KNN – Number of Neighbors Effect on Accuracy



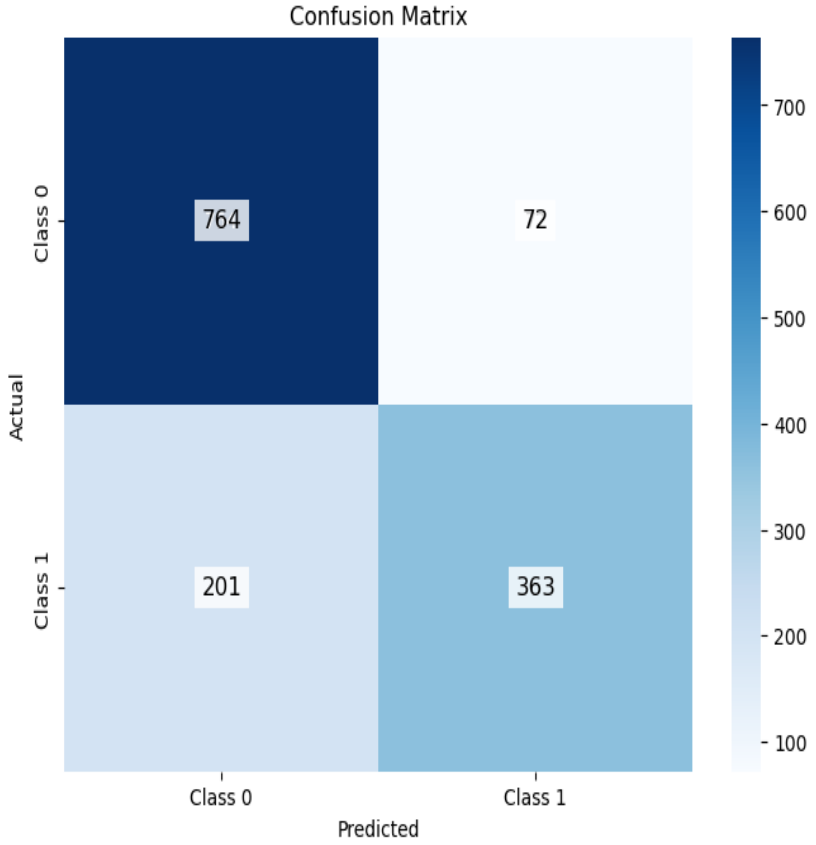
K-Nearest Neighbors

Trained Data Accuracy: 0.805
Test Data Accuracy: 0.740

- 1751 rows of data
- Target Value – W/L (Win/Loss)

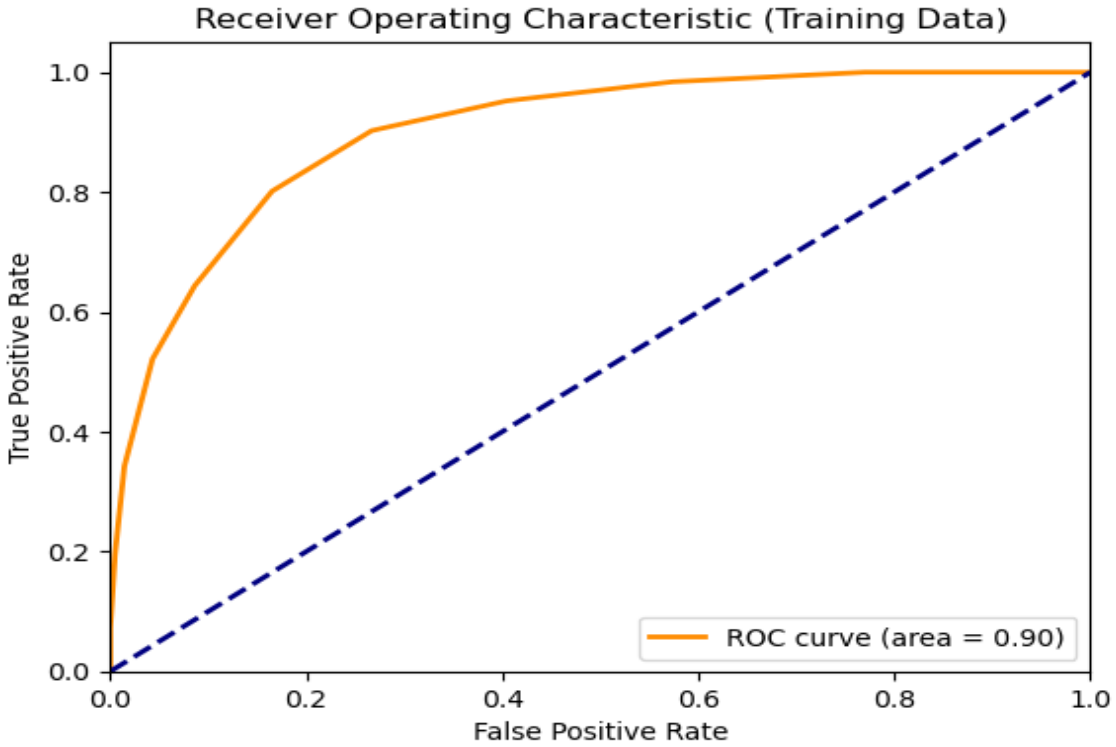
```
Feature names: Index(['Gm#', 'D/N', 'H/A', 'Opp', 'COL_at_bats', 'COL_ba', 'COL_hits',
       'COL_hr', 'COL_kk', 'COL_obp', 'COL_walks', 'Opp_at_bats', 'Opp_ba',
       'Opp_hits', 'OPP_HR_Column', 'OPP_kk', 'Opp_obp', 'Opp_walks'],
      dtype='object')
```

```
# Apply KNN
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train_scaled, y_train)
y_pred = knn.predict(X_test_scaled)
```



KNN X_trained Results

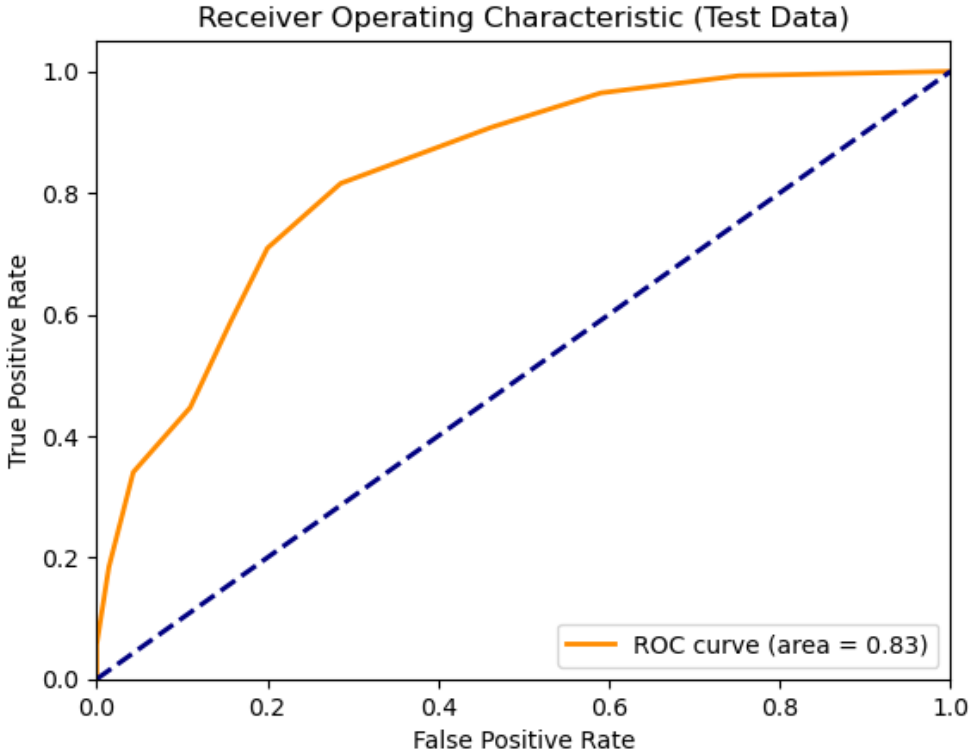
```
Training Accuracy: 0.805
Training Confusion Matrix:
[[764  72]
 [201 363]]
Training Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.91      0.85       836
           1       0.83      0.64      0.73       564

    accuracy                           0.81      1400
   macro avg       0.81      0.78      0.79      1400
weighted avg       0.81      0.81      0.80      1400
```



The model achieves an accuracy of approximately 80.5% on the training set. This indicates that the model correctly classifies about 80.5% of the instances in the training data.

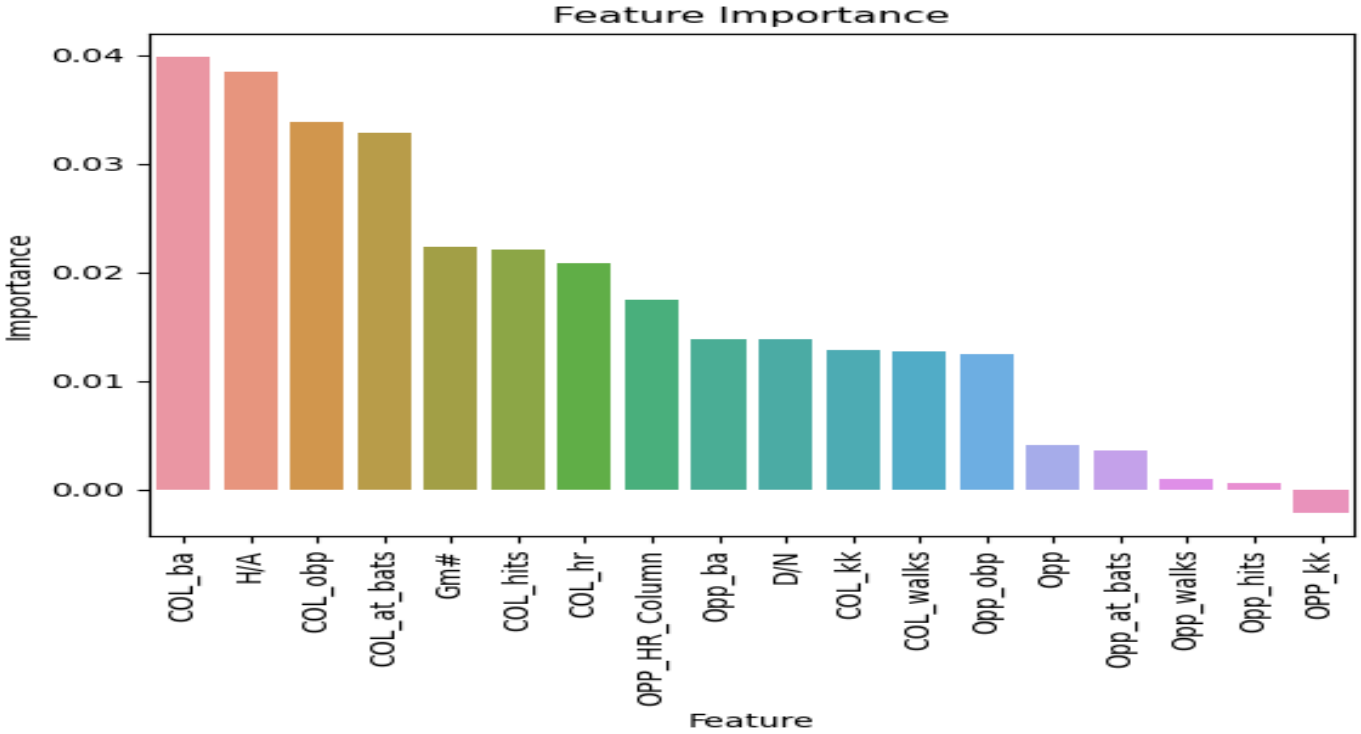
KNN X_test Results

```
Test Accuracy: 0.7407407407407407
Test Confusion Matrix:
[[177  33]
 [ 58  83]]
Test Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.84      0.80       210
           1       0.72      0.59      0.65       141

    accuracy                           0.74       351
   macro avg       0.73      0.72      0.72       351
weighted avg       0.74      0.74      0.74       351
```



The model achieves an accuracy of approximately 74.1% on the test set. This indicates that the model correctly classifies around 74% of the instances in the test set.

KNN Feature Importance



The top 4 features identified by using the permutation importance function were COL_ba, Home/Away, COL_obp, and COL_at_bats. The least werre Opp_hits and Opp_kk.

KNN Conclusion:

- Overfitting: The model appears to be overfitting, as it performs better on the training data than on the test data. This is evident from the higher training accuracy and slightly inflated performance metrics on the training set.
- Class Performance: The model performs relatively well for Class 0 but struggles more with Class 1, particularly on the test set. The reduced recall and F1-Score for Class 1 in the test set indicate that the model has difficulty generalizing to new data for this class.
- Generalization: The decrease in performance metrics from training to test data suggests that the model might not generalize well to unseen data, which could be due to the model being too complex, insufficient training data, or an imbalance between the classes.

Recommendations

- Address Overfitting: Implement techniques to reduce overfitting, such as regularization, pruning, or using a simpler model.
- Need to Improve Class 1 Performance: Explore methods to enhance the model's performance on Class 1, such as using class weights, resampling techniques, or feature engineering to better capture the characteristics of Class 1.
- Model Evaluation: Continue to evaluate the model on additional validation sets and potentially adjust hyperparameters to balance performance across different classes and data splits.

SVM (Support Vector Machines)

Description: Finds the optimal boundary to separate different classes.

Example: Classifies games as wins or losses based on complex performance metrics.

Key Takeaways: Handles non-linear relationships well; robust for high-dimensional data.

Overview

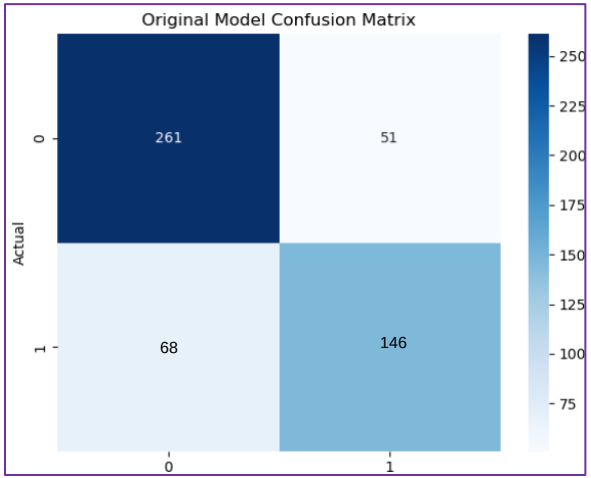
- The confusion matrix breaks down our model's predictions into four categories: **True Negatives** (Top Left Quadrant), **False Positives** (Top Right Quadrant), **False Negatives** (Bottom Left Quadrant), and **True Positives** (Bottom Right Quadrant).
- By examining these categories, we can see where the model is preforming well and where it is making mistakes, providing a clearer picture of its strengths and weakness.
- The heatmap we're about to see will show the performances of our model before and after tuning. The color gradients will help us quickly identify areas where the model has improved or where further adjustments might be needed.

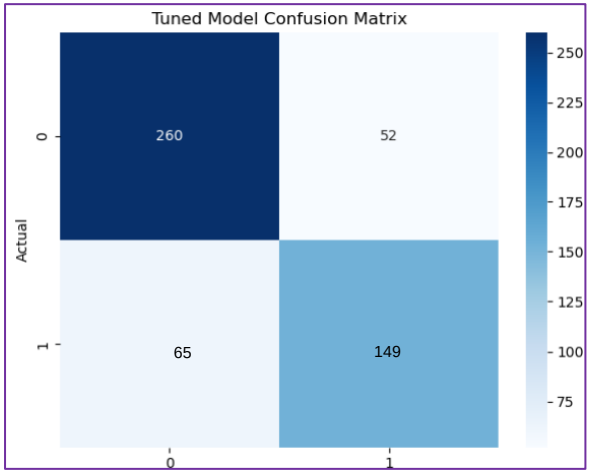
SVM – RBF

An **RBF** (Radial Basis Function) handles data that is not linearly separable, meaning it can't be separated by a straight line. Think of the RBF kernel as a magical way to stretch or twist the data into a new shape that makes it easier to find a boundary between different groups.

Key Metrics Summary:

- Details:
  - Original Model:
    - Accuracy: 77%
    - Precision for Losses (Class 0): 79%
    - Recall for Wins (Class 1): 68%
  - Tuned Model:
    - Accuracy: 77%
    - Precision for Losses (Class 0): 80%
    - Recall for Wins (Class 1): 70%



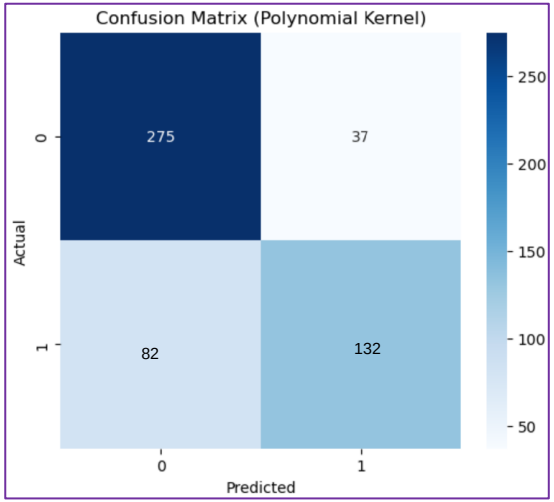


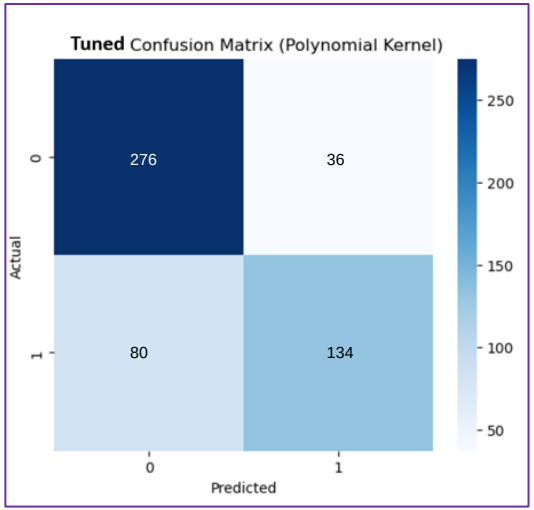
SVM – Polynomial

Think of the **Polynomial Kernel** as a way to draw curved boundaries of varying complexity to separate different groups of data. Instead of just looking for a straight line, it looks for a curved line (like a parabola or a cubic curve) that can better separate the data points into distinct categories.

Key Metrics Summary:

- Details:
  - Original Model:
    - Accuracy: 77%
    - Precision for Losses (Class 0): 77%
    - Recall for Wins (Class 1): 62%
  - Tuned Model:
    - Accuracy: 78%
    - Precision for Losses (Class 0): 78%
    - Recall for Wins (Class 1): 63%





Feature Importance

Most Important Features:

- Opp_at_bats" (Opponent At-Bats): The longest bar on the chart, indicating this feature has the highest importance in the Linear SVM model. This suggests that the number of at-bats by the opposing team is highly influential in predicting game outcomes.

- Opp_ba (Opponent Batting Average): The second most important feature, indicating that the opponent's batting average also plays a significant role in the model's decision-making process.

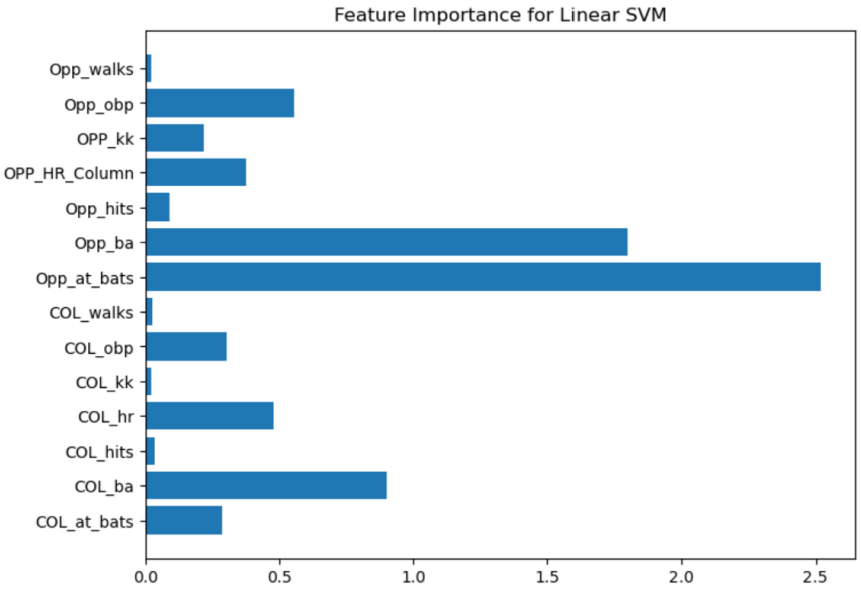
Moderately Important Features:

- COL_ba (Colorado Rockies Batting Average): This feature has a noticeable importance, showing that the Rockies' batting average is also a significant predictor in determining game outcomes.

- COL_hr (Colorado Rockies Home Runs) and OPP_HR_Column (Opponent Home Runs): Both features have some level of importance, suggesting that home runs, both by the Rockies and their opponents, contribute to predicting game outcomes.

Least Important Features:

- Features such as Opp_walks (Opponent Walks), COL_walks (Colorado Rockies Walks), and COL_kk (Colorado Rockies Strikeouts) have very small or negligible importance, indicating that they play a minimal role in the model's prediction.



Conclusion:

Model Perforamnce Comparison

Model Performance Comparison:

- Accuracy:
  - Both kernels achieved similar accuracy (77%-78%), showing solid performance in predicting game outcomes.
- Precision and Recall:
  - RBF Kernel: Best balance achieved with tuned parameters (C = 1, gamma = 'scale'), improving recall for Wins.
  - Polynomial Kernel: Moderate improvement in recall and precision for predicting Wins, with slight overall accuracy gain.
- Insights Gained:
  - Tuning Matters: Parameter tuning, while producing modest gains, helped optimize both kernels for their strengths—highlighting the importance of fine-tuning in machine learning models.
  - Kernel Selection: The choice of kernel affects the model's ability to generalize and balance between different performance metrics, suggesting the need to experiment with different kernels based on the specific data characteristics.

Wrap Up:

Conclusion

Our analysis of the Colorado Rockies' 2024 season combines a rigorous data-driven approach with a diverse set of machine learning models to predict future performance. By examining over a decade of game statistics, we identified key factors influencing outcomes and leveraged models like Decision Trees, Random Forests, Linear and Logistic Regression, SVM, and KNN to provide a robust framework for predictions. Each model offered unique insights—from the straightforward interpretability of Decision Trees to the advanced pattern recognition of SVM and KNN—allowing us to capture the complexities of the game. Our findings not only forecast potential win-loss scenarios but also deliver strategic insights for optimizing performance and decision-making. This comprehensive approach reflects our commitment to using data and technology to provide meaningful, actionable predictions that can guide the Rockies' path forward in the remainder of the 2024 season.

Key takeaways:

- Best scoring model: RandomTree Forest (0.820)
- Most impactful features vary per machine learning model
- Average team data not sufficient to predict future wins or losses
- Models may be useful for predicting other variables

Thank you!

Q&A

Sources

- Data was pulled from various sources using **pybaseball** package for python.
- Referenced tables located on:
  - www.baseball-reference.com
  - FanGraphs: https://www.fangraphs.com/
  - Baseball Savant: Statcast, Trending MLB Players and Visualizations | baseballsavant.com
- ChatGPT was referenced for building custom functions, debugging, and general guidance.

Appendix