Predicting Penalty Kick Models Using Machine Learning Models

Christopher Paucar
Adviser: Xiaoyan Li

May 1, 2023

Abstract

This paper details the design, implementation, and evaluation of a football (soccer in the U.S) penalty predictor that can be used in a myriad amount of ways. This can include teaching goalkeepers whether a penalty will go in based on the penalty kicker's foot and direction of the ball they shoot. Another way includes predicting penalty kicks for betting purposes, or simply trying to predict whether a penalty kick will go in live, putting features into a model and then comparing them to the user's guess. This project uses various machine learning models such as decision trees and random forest, to be able to offer the user a simpler way of being able to predict whether a penalty kick will go in or not. This application of a predictor can be used to predict penalty kicks live during nationally televised penalty kick shootout and even be applied for betting purposes of the user.

1. Introduction

On December 18, 2022, millions of people held their breath around the world. Argentina played the reigning champions France in the final of the FIFA World Cup in Qatar. Argentina was winning 2-0 with 10 minutes left to play, but in 2 minutes, French star Kylian Mbappe scored twice to tie the game at 2-2, where it went into overtime. In the 4 goals that were scored in the first 90 minutes, 2 of them were penalty kicks After the subsequent 30 minutes of play, the game was tied at 3-3, with one of those goals being a penalty. As a result, the game went to a penalty kick shootout, were, after a high level of suspense, Argentina won the shootout and the game, earning their 3rd World Cup.

Football also known as soccer, is the world's most popular sport. Two teams of 11 play against each other, and if the game ends in a draw, the teams will typically go to an extra 30 minutes of play.

It's important to note that goals can also come in the form of penalty kicks. If the game is still tied, then the game will go to penalty kick shootout, where the winner of this shootout wins the entire game. As a result, a lot of fans really care about how penalty kicks are taken because how well a penalty kick is taken can determine if the penalty goes in, which in turn determines which team will win. This all puts even more pressure on the players. Players who score are lauded as heroes, while those who miss are scapegoated and sometimes even abused.

I have played this game for as long as I can remember, and I've been on both sides, taking penalties, sometimes scoring and sometimes missing. I've also watched them as a fan, predicting on my own whether the shot would go in or not. However, what if there was a way to predict whether a player would score a penalty or not, based on certain features that one could see as the shot is being taken? What if one could make a machine learning model and predict the outcome, perhaps comparing the model's prediction with an avid soccer fan's prediction and compare it with what actually happened in the game? Such answers can have applications in fields such as betting, whether it be professional betting or just among friends. Another application is teaching young soccer players ideal features that will most likely result in penalty goal, and follow them. In the following report, I intend to answer these questions in the following manner. First, I will investigate similar works on this topic, and find differences between these other approaches and my. Second, I will describe both the approach I developed and the implementation that I used. Finally, I will evaluate and analyze my findings, in order to draw conclusions and think of future directions that I can take.

2. Background and Related Work

2.1. Background

The context this work will fit is in the prediction of penalty kicks in football. However, it's important to note some aspects of the data that will be collected.

Not all football games require penalty kick shootouts.

In football, not all of the games actually contain penalty kick shootouts. In fact, the majority of games actually **DON'T** end up in a penalty kick shootouts. The only types are games that do require one are elimination games, where a winner needs to be decided. Some examples of these games are World Cup Semi-Final and Final Games, among others. In these instances, there has to be a winner. The winner moves on to the next round (or win the tournament if it's the final), and the other team is eliminated. However, games that can end in a tie don't require penalty kick shootouts as there is no need to declare a winner. Examples of these include soccer games from a country's domestic leagues. If the game ends in a tie, then it ends in a tie, and no winner is decided.

In football, penalty kicks don't only occur in a penalty kick shootout

In football, penalty kicks don't only occur in a penalty kick shootout. Penalty kicks also occur when an attacking player is fouled in opposing team's penalty box, which is a 40.3 x 16.5 m box

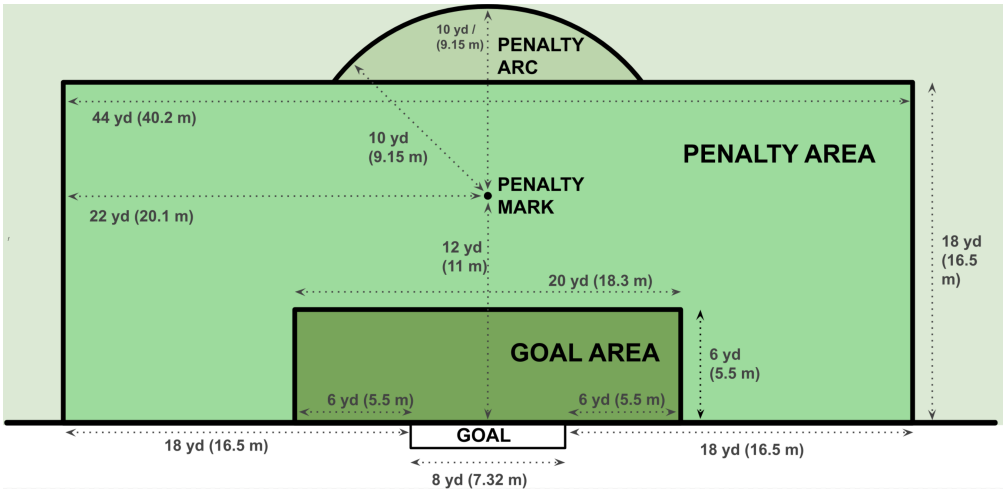
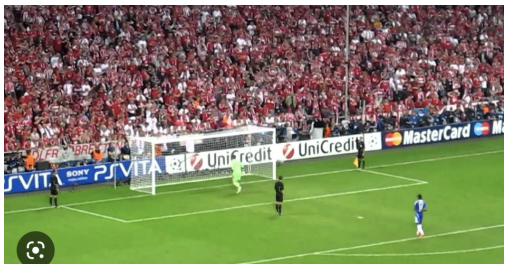


Figure 1: Diagram of Penalty Box [12]

near the opposing team's goalkeeper. These types of penalties should also be included when cleaning the dataset to train in a model. For example, in the 2012 UEFA Champions League Final between London-based Chelsea FC and Munich-based Bayern Munich, Bayern Munich star Arjen Robben took a penalty that was saved by the Chelsea goalkeeper. This penalty kick is a perfect example of a penalty kick that did **NOT** occur during a penalty kick shootout. Coincedently, that

football match ended up going into a penalty shootout.

(a) Arjen Robben's penalty against Chelsea, NOT part of the penalty shootout.



(b) One of the penalty kicks in the shootout, which happened in the same match.

Figure 2: 2012 UEFA Champions League Final between Chelsea FC and Bayern Munich, which shows a penalty kick [11] in a shootout and a penalty kick caused by a foul

Most penalties in football are scored successfully.

Another aspect that is important to note is the high rate of penalty success for the kicker of the penalty kick. For example, the data used by the researchers in the article [4] **The Interplay of Goalkeepers and Penalty Takers Affects Their Chances of Success**, which consisted of FIFA World Cups and UEFA European Championships from 1984-2016, contained 289 successful penalty kicks and 106 unsuccessful penalty kicks, for a total of 398 penalty kicks. This means that 73.2% of the penalty kicks went in, and 26.8% that missed. In another research paper, named **In-match penalty kick analysis of the 2009/10 to 2018/19 English Premier League competition**, the authors note how through the research they conducted, the penalty kick success rate [8] in professional soccer leagues range from 70% to 85%. This means that, when collecting a dataset of penalty kicks from a particular season in football, most of them would go past the goalkeeper, which needs to be taken into consideration in my approach and implementation stages.

With this background information now known, it's important to make comparisons of my project with respect to similar works conducted by others. Below are 2 examples of related works and their comparisons with the work from this paper.

2.2. Related Works

The idea of predicting penalty kicks has been applied in other fields as well. However, the features and methods used to predict penalty kicks are different than the features and methods that were used in the following report. The following were examples of models trained to predict penalty kicks

1. **ROBOKEEPER: The Robotic Goalkeeper**
2. **Predicting Football Penalty Directions Using In-Match Performance Indicators**

In "ROBOKEEPER: The Robotic Goalkeeper", the ability to predict penalty kicks comes in the form of a robot inside of a soccer goal, meant to physically move based on certain features in collects. It's in this process of feature extraction where my project differs. I will use features extracted from past penalty kicks, and after performing feature engineering and EDA, will use this to train my model. However, in the case of "ROBOKEEPER", the features are extracted in the following manner.

Two cameras located on the "eyes" of the record around 90 images per second, and these eyes follow the motion of the soccer ball. It's important to note that the soccer ball in this case is chosen based on its color, as the robot eyes work best when it has contrasting color [1]. In addition, the sensors that the robot uses are located on the ball too, making it necessary to equip the soccer ball beforehand. Afterwards, the robot uses image processing to be able to make a decision as where to dive in order. Once this decision is made, it will use dynamics and planning to physically move the robot.

While this form of predicting penalty kicks is very accurate due to the intricacy that comes with real-time image processing, I can see drawbacks to this form of prediction. First of all, in a real football environment, it's unfeasible to have the goalkeeper and soccer ball to be geared up with significant sensors, especially with the fact that there are already sensors present, such as to predict offside calls. In addition, and this is more specific to achieving the goal of this paper, since this is a robot with various sensors, one needs space to setup this type of predictor and a special soccer ball, something that isn't necessarily readily available. This also makes such a predictor difficult to be portable to the average football fan.

In "Predicting Football Penalty Directions Using In-Match Performance Indicators", the method that the authors, Lotte Bransen and Jesse Davis, used to predict penalty kicks was through in-match performance indicators. Their goal was to provide goalkeepers with predictions as to where to dive. In order to accomplish this goal, first they gathered their penalty kicks from a dataset provided by SciSports, which contains match statistics from 25,847 matches [6]. Next, they extracted the features they wanted, which are listed below

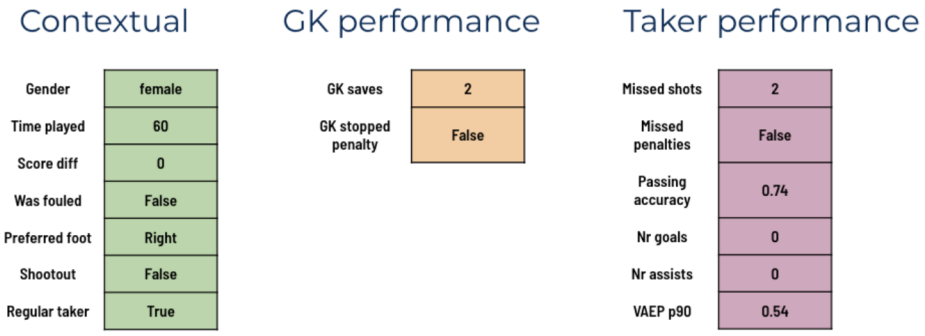


Figure 3: Diagram of Features [6] Used By "Predicting Football Penalty Directions Using In-Match Performance Indicators."

With these features extracted, they then had to get features that measured player performance, which includes them using VAEP [6], or Valuing Actions by Establishing Probabilities, which values every on-the-ball action based on its impact on scoring and condeding a goal in the near future [6], and determining expected values per player. Once this was done, they trained the model using both an XGBoost and Explainable Boosting Classifier to “estimate the probabilities of the players shooting either on the goal’s left, center, or right.”[6]

The difference between my paper and the work done by Bronsen and Davis is their use of in-game metric statistics. They used features that I didn’t such as the kicker’s passing accuracy and missed shots, or the goalkeeper’s saves, among others. However, this is a major drawback if I were to use this on my dataset. Since these features are in-game, it would be most accurate for penalties that are taken late in the game, or in penalty shootouts, because a large amount of time has passed for features to be collected. Penalty kicks shootouts typically occur after 120 minutes, and any penalties

taken late in a football match also have time to collect these features. However, what if there was penalty early in the game, such as the 5th minute, or 20th minute. There wouldn't be enough time to collect in-game features.

3. Approach

The authors of these previous related works have offered solid methods of predicting penalty kicks. However, as stated previously, their approach contains some caveats that I want to address and intend to fix as I developed my approach in my paper. Subsequently, my approach involve the combination of the following aspects.

1. **Moving away from non-penalty-kick-shootout penalty kicks**

   As mentioned in my background, penalty kicks in football don't only occur in penalty kick shootouts. Most football games don't have penalty kick shootouts, and in these types of games, this means that penalty kicks would come of the form of fouls within the penalty box. However, it is still very important to predict these penalty kicks because they can affect the final score of a football match. For example, team A ends up tied against team B in a league game, which isn't an elimination game, and therefore has no penalty kick shootout. However, team A could've won the game had they scored a penalty kick. Team A would've won, but because of a missed penalty, they ended up tied. As a result, when looking at datasets to use, I will have to use datasets from various league games, as these non-elimination games will contain even more types of penalties that will be used as datapoints in training my models in preparation to form a prediction. This represents a difference from past works such as in "ROBOKEEPER: The Robotic Goalkeeper", whose datasets are live penalty kicks detected by sensors [1] and processed using image processing and computer vision.

2. **Feature Selection**

   I want to use features that are going to be easily visible to someone watching a live soccer game. However, what is really important is that I want my approach to use features that are already collected and noted. Why? The reason is because, as is the case in the paper "Predicting Football

Penalty Directions Using In-Match Performance Indicators", the issue with in-game features is that when penalty kicks occur early in the game, there isn't enough time to get meaningful features. In fact, they mention this towards when discussing possible improvements [6], where "at the start of the match we have lower indications of a player's in-match performance which causes our model to perform worse for early-match penalties."

3. **Model Diversity**

   As I've learned throughout the semester, there are a multitude of machine learning models that can be used, from logistic regression, to gradient boosting. Each model has a tradeoff, performing better than others in some metrics, and this is true after performing hyperparameter tuning. As can be seen in "Predicting Football Penalty Directions Using In-Match Performance Indicators", the authors train their data on an XGBoost[6] and an Explainable Boosting Classifier, whereas in "ROBOKEEPER: The Robotic Goalkeeper", there is no mention of the type of model [1] that they use, only how they features collected from the sensors and cameras allows the robot to predict where to go. Therefore, I want to be able to train my data on multiple models [3], because, as stated by authors such as Sekeroglu et al., the tradeoffs in different models can cause differences in the models' performance, which is affected even more by the type of dataset that I will use. As a result, I want to be able to train my model on more models than those explicitly stated that were used in my Related Works.

4. Implementation

4.1. Step 1: Datasets

Before being able to perform any sort of analysis on having a machine learning model predict whether a penalty kick would go in, first I need to get data. Given the limitations in my related works, I needed datasets where the features wouldn't be too reliant on the time the penalty kick occurs. I also wanted these features to be able to learn from previous penalty kicks. For example, from penalty kicks 3 years ago, or 5 years ago. Finally, I wanted penalty kicks from both elimination and non-elimination games. As a result, I collected data from 4 datasets.

1. **Exploring EPL Penalty Kicks Dataset**

   This dataset, which I downloaded from Kaggle contains penalty kicks [2] from the 2016-2017 English Premier League, which is the name of the domestic league in England. This dataset contains 106 rows, where each row represents a penalty kick, and it contains 13 features. This dataset is used because since this dataset is from penalties in a domestic league, these games occured in non-elimination games.

2. **World Cup Penalty Shootouts**

   This Kaggle dataset contains penalty kicks[9] that are involved in penalty kick shootouts from all World Cups between 1982 and 2018. This dataset contains 279 rows, where each row represents a penalty, and 9 features. This dataset contains penalty kicks shootout, where the winner moves to the next round, and the loser is eliminated.

3. **Penalty Statistics 2019-2020**

   This Kaggle dataset is a large dataset that contains smaller subsets of penalty kicks [5], where each subset are penalty kicks from various competitions, such as the Spanish domestic league, the French domestic league, the UEFA Champions League, the Italian domestic league, and the German domestic league. This dataset contains 494 datapoints, with 18 features.

4. **Missed Penalty Dataset**

   For reasons that will become clear later on, I needed penalty kicks where the kicker ended up

missing the shot. As a result, I had to look through YouTube and write down the necessary features of each penalty that I wanted to work with. In the end, this dataset contained 80 rows with 3 features.

4.2. Step 2: Preprocessing

After getting my datasets, next I wanted to preprocess my data. This involved using Python libraries such as *pandas* and *numpy* to be able to get rid of any features that I wouldn't use. This included features such as the names of the players, the team name, the date of the game, among others. The features that I decided to use were the features in Table 1. Afterwards, I eliminated any empty rows

Table 1: Table of Features

Desired Features
Dominant Foot: Which foot (Left/Right) the kicker uses to kick the penalty kick.
Kicker Direction: The direction the penalty shot goes (Left/Center/Right)
Keeper Direction: The direction the goalkeeper dives (Left/Center/Right)

that were present in my datasets. Once this occurred, I combined all my prepocessed data into a single dataset, which would be the one that would be used to train and test my models. This dataset consisted of 503 rows and 8 columns that represent my features.

4.3. Step 3: Feature Engineering and EDA

The following steps occured before combining my datasets to the my model input, but I felt it more relevant to talk about here since it regards my features. After talking to my advisor in regards to the small amount of features, I decided to One Hot Encode my features, which changed my categorical data from L's, R's, and C's 1's and 0's, and allowed me to increase the number of features to 8 features. After combining the datasets, the next step was preprocessing. After receiving guidance from Professor Li, I used One Hot Encoding to transform categorical data into numerical data. In addition, this allowed me to breakout the following features into 8. Mainly, I had the following features.

With my model input dataset now finalized, I now moved on to performing EDA, or Exploratory

Data Analysis, and looked at any relationship that I could perform. The first thing I did was split my data into a train and test split by the standard 80-20 split, and then followed this up with another 80-20 split, this time splitting my training set into a training and a validation set, which would become useful when performing hyperparameter tuning. This will help in reducing overfitting, as will be explained later. Now, I performed EDA on my training data, and came to the following conclusion: my data was imbalanced. This imbalanced nature goes back to my background, where the success rate [8] for scoring a penalty kick is around 70%. This means that, in my dataset, I should have more penalty kicks whose label is that it goes in compared to those that don't go in, and the following Figure 4 shows this imbalanced nature.

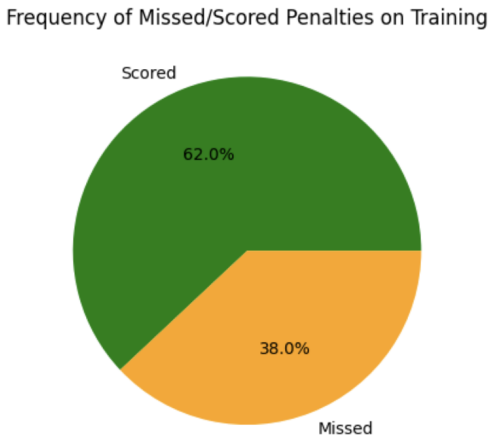


Figure 4: Diagram Showing Imbalanced Nature of my Dataset

The reason this has to be handled is because, when a model is given an imbalanced dataset, the machine learning model will be biased towards the majority class, which in this case are predicting penalty kicks that go in.

In order to handle the imbalanced nature, I took the following steps. I first gathered unique datapoints/penalty kicks where the kicker ended up missing the kick, which is where the dataset with missed penalties was used. It was actually much more imbalanced before, closely to the numbers given by "The Interplay of Goal-keepers and Penalty Takers Affects Their Chances of Success" [4] and "In-match penalty kick analysis of the 2009/10 to 2018/19 English Premier League competition" [8], which were 70% of the penalty going in and 30% of the penalty not going in.

These penalties were from the 2022 World Cup and from competitions such as Copa America 2021, Euro 2021, Copa America 2019, and Copa America 2016.

The next step was performing Random Oversampling, which is a process where I randomly select examples from the minority class and add them to the training data until a balanced dataset is achieved. This means that examples are duplicated from the minority class, effectively "oversampling" it to match the size of the majority class. I performed random oversampling on my training data, and as a result, I had a training with 199 penalties going in and 199 penalties not going in. With my training data now balanced, I could now move on to model selection.

4.4. Step 3: Models

4.4.1. Model Selection

One of the biggest differences between my work from previous works is the diversity of models that I trained, validated, and test my data on. When deciding which model to use, I wanted to use models that are general and common, as well as being commonly used in research by machine-learning enthusiasts. With this in mind, the machine learning models that I chose to use includes

- **Decision Tree**

  The Decision Tree Classifier is a type of supervised machine learning model. Here, a dataset is split into multiple sets, based on important characteristics [10]. It is simple and has a relative short raining time, which is helpful since I want to train and compare the metrics of multiple models.

- **Logistic Regression Classifier**

  This model was chosen due to its popularity in use among machine learning engineers. Similar to decision tree, it is simple and takes a relative short amount of time to train, and the output of the model is easy and quick to interpret.

- **Support Vector Machine Classifier**

  This machine learning model was chosen for various reasons. First, it's simple to use and although it doesn't have the best training time, with my small number of features and numerical data after One-Hot-Encoding, this model would work well. This is especially true given that it is robust to

overfitting, which is prone to occur in my project given that I performed random oversampling to get the minority class equal to the majority.

- **Random Forest Classifier**

  Random Forests classifiers consists on many decision trees that function on small subsets of data that the model is given. The reason I chose this model is because, similar to SVM's, it's robust to overfitting, which could occur given the Random Oversampling that was performed.

- **Gradient Boosting**

  The reason I chose this machine learning model was because of its ease at interpreting predictions that it performs, as well as being able to handle non-linearity.

All these models were trained on my oversampled training data. Once trained, I next performed hyperparameter tuning, in order to be able to enhance my performance. I used my validation set in order to tune my hyperparameter tuning, with the help of the **Grid Search CV** library. Once this was done, then, with the tuned models, I went on to train the model and the combined training and validation set, and compared the model's predictions with that of the test. During these steps, I got various metrics such as F1 and ROC scores, and performed evaluation of my models in order to draw conclusions about how successful my model was in predicting penalty kicks with the given features.

4.4.2. Python Libraries

The following are the Python libraries that were used are where in the implementation process they were used for.

- **Pandas**: This library was used to import my datasets from Kaggle and Google Spreadsheets, as well as data manipulation and processing.
- **Numpy**: This library helped me manipulate my datasets in conjuction with the **Pandas** library.
- **sklearn.preprocessing**: This library was used when performing One Hot Encoding to one-hot my original features.
- **Counter**: This library was used to import Counter, which was used when counting the number of penalty kicks that went in and didn't go in.

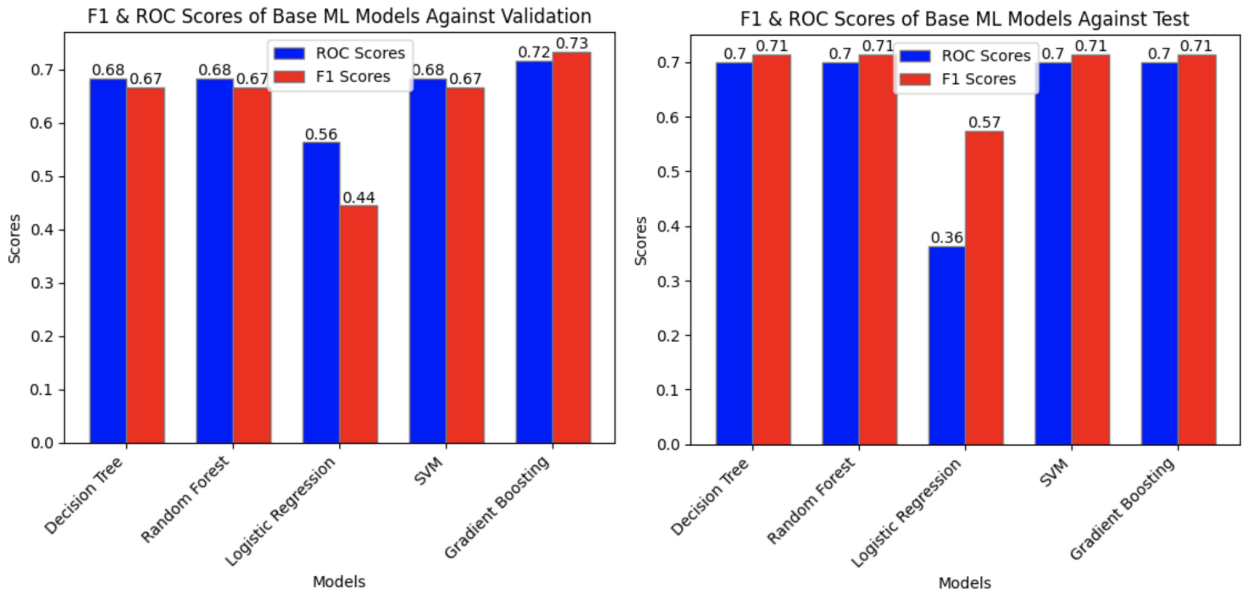
- **RandomOverSampler**: Taken from the imblearn.oversampling, this library was used when balancing my training dataset.
- **sklearn**: This library was used to create the machine learning models, implement GridSearchCV to tune my hyperparameters, and analyze my findings through the metrics provided.
- **matplotlib**: This library was used for plots, such as pie charts and bar graphs
- **seaborn**: This library was used for creating the heatmats for my confusion matrices.

5. Evaluation

5.1. Step 4: Iterative Analysis of Metrics

Continuing with the steps started in the Implementation section, I analyzed my metrics in order to be able to analyze my findings and edit my models depending on my analysis. First, when evaluating the performance of my models, I decided to use F1 score and ROC score. This is because, initially, my dataset was imbalanced in favor of penalties going in. As a result, even though I performed random oversampling, I decided to use F1 and ROC scores, as they are more useful in handling imbalanced datasets. Figure 5 reflect the ROC and F1 scores of the models used to predict penalty kicks with the features from Table 1:

The bar graph on the left shows F1 and ROC scores based on how the base model's(without hyperparameter tuning) performance against the validation set, and the bar graph on the right shows the same scores with the tuned model's performance against the test set. The Decision Tree, Random Forest, and SVM actually performed extremely similar to each other, basically identical when comparing my train against my validation test and my train against my test set. And in fact, they actually increase, from an F1 score of 67% to an F1 sore of 71%, and an ROC score from 68% to an ROC score of 70%. These could be insight to these models preventing overfitting, since the model is showing higher percentages on the test data than when comparing it to the validation set. As a reminder, overfitting occurs when a machine learning model learns too well the training set, and as a result, when it makes its predictions, it does really well on the train and validation set, but



(a) Bar Graph Showing F1 and ROC Scores Against Validation

(b) Bar Graph Showing F1 and ROC Scores Against Validation

Figure 5: Bar Graphs Showing Metric Against Validation Set and Test Set

poorly against the test set. Since the test set's F1 and ROC scores are actually higher, this means that it's generalizing well.

When comparing the F1 and ROC scores of the remaining models, the performance is not the same. In the case of Logistic Regression, the ROC scores goes from 56% down to 36%, and the F1 scores goes up from 44% to 57%. Not only is this much more unstable than the previous 2 models mentioned, but its performance is not as good as the previous 3 models mentioned. As a result, I decided to ignore this model.

In the case of Gradient Boosting, the drop in both the F1 and ROC score from 73% and 72% to 71% and 70% respectively. This slight drop in the F1 and ROC score could be a sign of overfitting, since the scores slightly drop when comparing it against the test. Now, with the scores analyzed, the next step would be to look at the confusion matrices, especially since these could give insight into potential errors and the error analysis that comes afterwards.

5.2. Error Analysis

As can be seen in Section 10.1 of the Appendix, I have the confusion matrix and ROC curve of the Base Random Forest (Figures 7 & 8 respectively), and the confusion matrix and ROC of the Tuned Random Forest Model Against the Test Set (Figure 9 & 10 respectively). The confusion matrix shown in Figure 9 is shown below for reference.

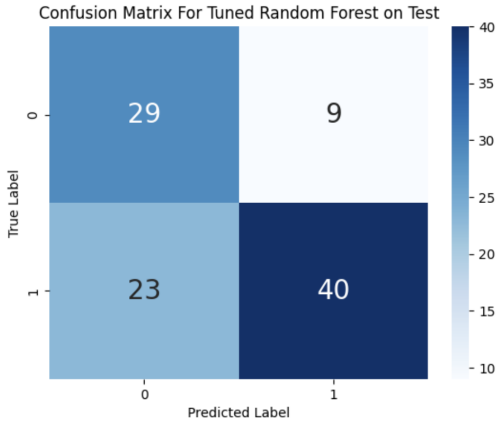


Figure 6: Confusion Matrix for Tuned Random Forest on Test Set

Looking at Figure 9, the model on 29 occasions, the model predicts a penalty shot going in correctly 29 times, while it predicts a penalty shot missing correctly 40 times. However, it predicts the a penalty shot goes in when it actually didn't go in 23 times, and the model predicted that a penalty didn't go it when it actually went in 9 times. How can this be? One way for the relative large number of false positives is through certain senarios that can occur in a penalty. Consider the following example.

A football player is about to take a penalty shot. The kick ends up kicking it to the left side of the goal. However, the goalie also dives to the same left side that the ball was being kicked. The model could be interpreting this as , "The ball and the goalkeeper are heading in the same direction, so the goalkeeper has a high chance of saving the penalty". However, the penalty goes it, whereas the model predicts that the goalie saved the shot. This is an example of a false positive.

This actually occurs various occasions within my dataset, where the model thinks that, since the

ball and the goalkeeper go in the same direction, it should save the penalty kick. One way I can try to minimize this error is to add the following features

- **Placement**

  In a penalty kick, the ball's direction and the goalkeeper's direction being the same doesn't necesarilly mean that the penalty kick will be saved. If the shot is placed well, such as in the corners and close to the goal posts, then even if the goalkeeper dives in the same direction as the goalkeeper, then the ball will go in, and the kicker scored the penalty. Adding this feature will hopefully minimize these types of errors.

- **Power**

  Powerful shots also have an impact too, as powerful shots can either mean penalties that are shot off target, or, if placed well, virtually impossible for the goalkeeper to save. In addition, not all penalty kicks are powerful, as a well placed shot can also go past the goalkeeper.

The goal of adding these features is not only to reduce error in the models, but to also see if F1 and ROC scores and increase across all models.

The confusion matrices for the decision tree is identical to the confusion matrices show in Figures 14, 22, and 26 in Sections 10.2, 10.4, and 10.5 of the Appendix respectively. Similar to Figure 5, the Logistic Regression Confusion Matrix shown in Figure 18 of Secion 10.3 of the Appendix shows again that the Logistic Regression Classifier seems to be doing worse than all other models.

5.3. ROC Curves

Another metric I used was the ROC score, and I used visuals of ROC Curves in conjunction with confusion matrices to be able to perform error analysis. An ROC (Receiver Operatating Characteristic) Curve is a curve that plots the rate of True Positives against the rate of false positives. As can be seen on Slide 20 [13] of Professor Li's "Model Training and Evaluation 2", the closer the plot is towards the top left hand corner (closer to the point (0,1)), the better. If the ROC curve falls on the line $y = x$, then the model is no better than if it were to randomly guess. Take, for example, the ROC Curve of Logistic Regression when comparing it to the Test: Notice how the ROC curve

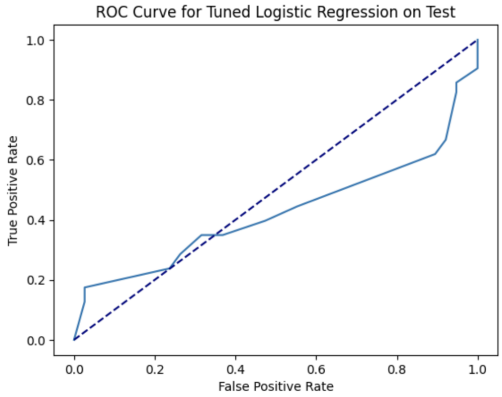


Figure 7: Confusion Matrix for Tuned Logistic Regression on Test Set

for this model falls almost along the dotted equation $y = x$ in certain locations, and other times it falls below it. This means that this model is no better than random guessing, further showing the need to steer clear of logistic regression. Now consider the ROC Curve for the Tuned Decision Tree Against the Test. This ROC Curve is actually tending towards the top left hand corner, which means

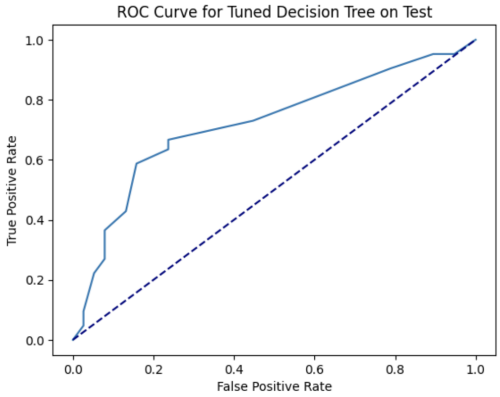


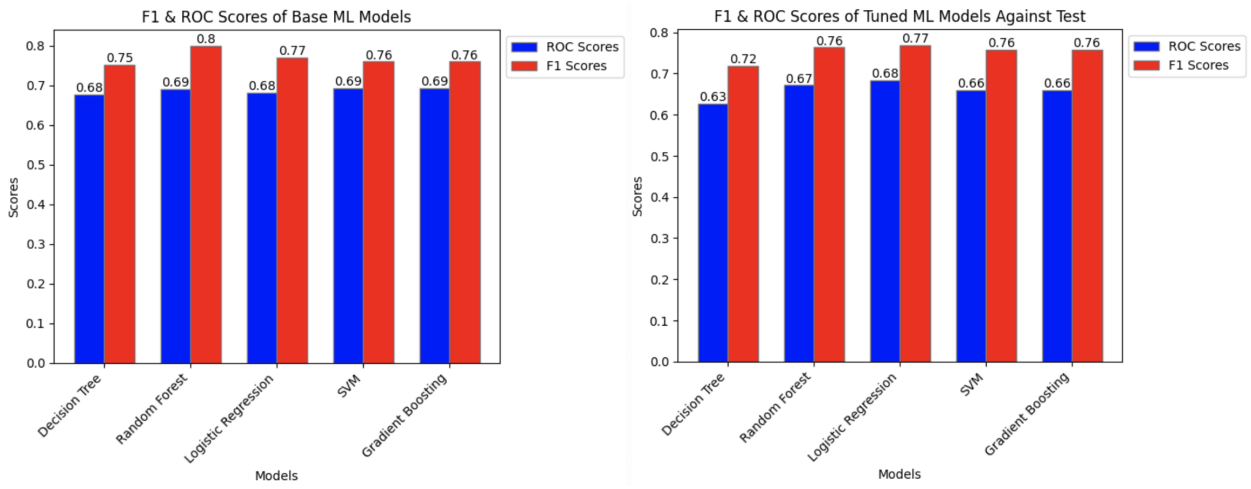
Figure 8: Confusion Matrix for Tuned Random Forest on Test Set

that it's better at predicting when penalty kicks truly go in and when they truly don't. With this is mind, I want to re-evaluate my metrics after implementing my new features.

5.4. After Adding New Features

The first thing to look at, similar to analyzing the evaluation metrics before the addition of new features, I will look at the F1 and ROC scores of these models, in the form of a bar graph.

Unlike in Figure 5, the addition of the **Placement** and **Power** features seem to have caused very slight overfitting, as can be seen by the before and after F1 and ROC scores. When comparing the



(a) Bar Graph Showing F1 and ROC Scores Against Validation

(b) Bar Graph Showing F1 and ROC Scores Against Validation After New Features

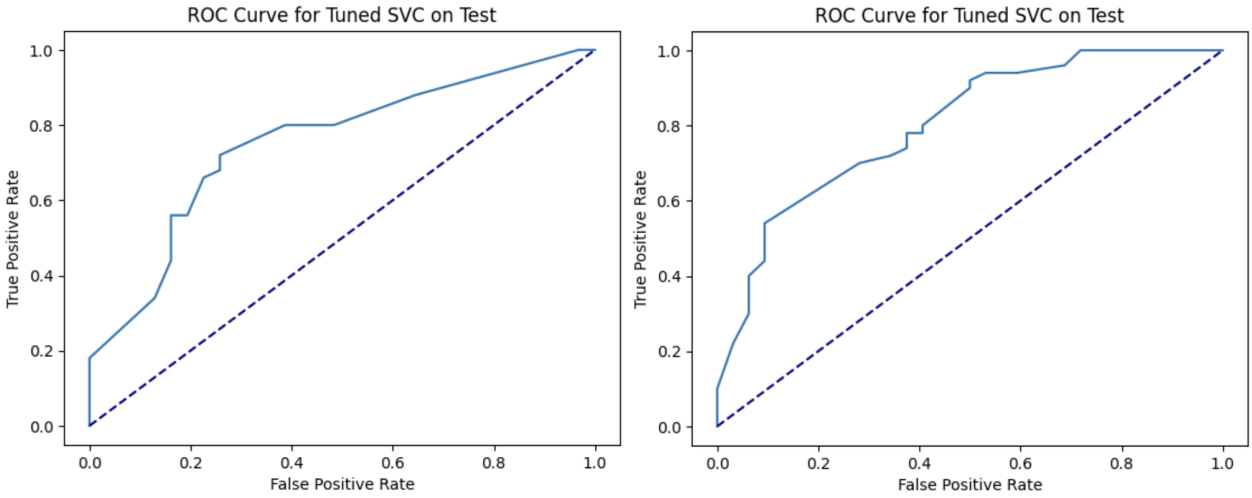
Figure 9: Bar Graphs Showing Metric Against Validation Set and Test Set After New Features

corresponding scores, most of them seem to either drop after hyperparameter tuning and comparing it to the test, or stay the same. However, there is one significant difference. The addition of these two features helped the Logistic Regression Classifier a lot, rising to a post-tuned F1 score of 77% and a post-tuned ROC score of 68%. In fact, this is the highest score out of all the other models, which is striking. This could occur due to the fact that it doesn't perform as well as other models with a small number of features.

5.5. ROC Curves

Let's first analyze the ROC Curves. From Section 10.6 until Section 10.10 of the Appendix, Figures 34, 38, 42, 46, and 50, which are the figures for each corresponding model against the test set, the ROC curves have improved, from slightly, to significantly. For instance, Figure 42, which shows the ROC Curve for the Tuned Logistic Regression on the Test, shows significant improvement, as the curve no longer falls on the line $y = x$, which is reflected by the ROC score increases in Figure 9. For the rest of the ROC Curves, when compared with the ROC Curves of the models before the addition of these features, the curves tend to hug the top-left corner over a larger range of False Positive Rates. For example, when comparing the ROC Curves for the Support Vector Classifier, as shown in Figure 10, before and after adding the new features, one can visually see how much closer

the curve hugs the left-hand corner, and how it tries to stay far from $y = x$.



(a) Bar Graph Showing F1 and ROC Scores Against Validation

(b) Bar Graph Showing F1 and ROC Scores Against Validation After New Features

Figure 10: Bar Graphs Showing Metric Against Validation Set and Test Set After New Features

5.6. Confusion Matrices

The confusion matrices on all of them, as can be seen in the Appendix, Figures 34, 38, 42, and 46, and 50, show the models primarily classifying the penalties as unsuccessful, or in other words, the player missed. This could be because of the new features having the model learn too many occasions of when the player missed the penalty. This is a great example of why to look at both the ROC curves and confusion matrices in conjunction. From just the ROC curve, it appears that the model has done much better. For example, with the **Placement** feature, the occasions of penalty kick direction and goalkeeper direction being predicted as missed is now more correctly handled. However, this addition in features, as can be seen in the confusion matrices, caused more penalties to be predicted as not going in, when it truly did.

6. Conclusion and Future Work

6.1. Conclusion

One of the biggest conclusions that I can draw is that, in order get more accurate and precise findings, it would be hepful to use more features. Whereas the features I chose to use to predict penalties are definitely important, there are definitely more important features. Similar to the paper "Predicting Football Penalty Directions Using In-Match Performance Indicators", how a football game goes for a penalty kick taker is extremely important in addition to be able to detect the dominant foot, the penalty kick's direction, and its placement, among the other features used.

Additionally I learned that placement in penalty kick shootouts is extremely important, especially after looking at the confusion matrices of the models before including that feature. Speaking from my own experience, placing the football in the correct spot is really important in order to have a high chance of making the penalty, and it was interesting to see this concept interpreted by the models that I used.

6.2. Limitations

There were some limitations in regards to this project.

- **Human Bias**

  Human Bias was a limitation in this paper in the form of the new features that were added. This is because in order to acquire these features, I looked through YouTube and wrote down placement and power as binary numbers (1 if it was placed well/powerful, 0 if not placed well/weak). I relied on my knowledge of playing football for so many years, but this, inherently, introduced bias into my project. Different people who were to watch this penalty clips could have different opinions, and for features such as power, it's hard to detect because it would require having a sensor on the soccer ball, which could make it heavier and affect the game itself.

- **Detecting Fakeouts**

  One of the strategies to score is to fakeout the goalkeeper, which means to trick the goalkeeper in

going one direction, but the kicker kicks it in the other direction. It's a strategy that many football players use, myself included. However, this is not accurately detectable without tools such as sensors that can detect features such as the location of the eyes, the angle of the kicker's foot, the angle of the waist, among others.

- **Rule 14**

  Rule 14 was a rule in football implemented after the 2018 World Cup, which states that the goalkeeper must keep at least one foot on or behind the goal line. This is important because before, goalkeepers were allowed to get closer to the penalty kicker, reducing the size of the goal in the eyes of the kicker. This also gave the keeper a head start in diving. What this meant was that sometimes a kicker might place the ball well, but because the keeper is closeby, the keeper didn't have to dive far in order to save it. With this being disallowed after 2018, however, it should more accurately reflect features such as placement, among others.

6.3. Future Work

In terms of future work, there are 2 ways I see this work being expanded upon in the future.

- **Addition of More Features**

  The reason I chose to go with the features listed in Table 1 was because these features could be collected from past penalty kicks without worrying about time, as in the paper regarding in-game metrics. However, in order to get higher-performing models, it would be useful to collect even more features. I added 2 features, predominantly placement and power, but more features could enhance the performance of my models. The addition of new features can also help me explain and improve the fact that the models heavily predict the penalty not going in, when it really did.

- **Use of Computer Vision and Sensors**

  Another way my work could be expanded is through using sensors and computer vision to be able to capture the necessary features, and in fact, this could more accurately predict whether a penalty shot would go in. For example, instead of just looking at the features from Table 1, there could be sensors to would allow the detection of the location of the eyes, the angle of the shoulders with

respect to the foot, the angle of the kicker's feet, etc. An example of such detections is shown below.

Figure 11: Diagram Sensor Detections of Body Angles Right Before The Kicker Kicks The Ball [7]

These features would be gathered by sensors and computer vision software, two concepts that are beyond my current knowledge in computer science.

7. Acknowledgements

First, I would like to acknowledge God, for giving me strength and wisom throughout this whole process. There were times where I was confused and tired, but through His help and guidance, I am here, about to submit my Independent Work and a piece of writing that means a lot to me. Next, I would like to thank my family, for being there in the tough times, for everything they have done for me, for loving me, and teaching so much value and honor. Next, I would like to thank Professor Li and Meet Patel, for guidance me through my first ever piece of Independent Work, of which I feel very proud of to show to my family and friends. I also want to thank Camila Vasquez and Natalie Reptak, for the fun times we had not just during the seminar, but for the past years that I've know you all. I'm truly blessed to call you my best friends, and for the fun times we had both during and outside of the seminar. And lastly, my seminar classmates for the feedback given during the presentations, which helped me carve a road in producing my presentation video.

8. Ethics

8.1. The Princeton Honor Code

This represents my own work in accordance with University regulations.

/s/ Christopher Paucar

9. References

References

[1] 4attention GmbH Co. KG, "Robokeeper: The robotic keeper," https://www.robokeeper.com/en/facts.html, 2007, accessed: 2023-2-10.

[2] Alexey Popov, "Exploring epl penalty kicks dataset," https://www.kaggle.com/code/apopov41/exploring-epl-penalty-kicks-dataset, accessed: 2023-2-03.

[3] a. K. T. B. Sekeroglu, K. Dimililer, "Student performance prediction and classificiation using machine learning algorithms," in *Proceedings of the 2019 8th International Conference on Educational and Information Technology*, ser. ICEIT 2019. New York, NY: Association for Computing Machinery, 2019, pp. 7–11.

[4] S. K. Benjamin Noel, John van der Kamp, "The interplay of goalkeepers and penalty takers affects their chances of success," *Frontiers in Psychology*, vol. 09, 2021.

[5] Emilie Richard, "Penalty statistics 2019-2020," https://www.kaggle.com/datasets/emilerichard/penalty-statistics-20192020, accessed: 2023-2-13.

[6] Lotte Bransen, "Predicting football penalty directions using in-match performance indicators," https://analytics.scisports.com/research/penalty_predictor, accessed: 2023-2-10.

[7] A. P. R. A. R. d. S. T. F. B. Luiz Vieira, Paulo Santiago, "Automatic markerless motion detector method against traditional digitisation for 3-dimensional movement kinematic analysis of ball kicking in soccer field context," 2022.

[8] W. K. Michael Horn, Simon de Waal, "In-match penalty kick analysis of the 2009/10 to 2018/19 english premier league competition," *International Journal of Performance Analysis in Sport*, vol. 21, 2021.

[9] Pablo L. Landeros, "World cup penalty shootouts," https://www.kaggle.com/datasets/pablollanderos33/world-cup-penalty-shootouts, accessed: 2023-2-10.

[10] Sunil Ray, "Commonly used machine learning algorithms," https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/, 2023, accessed: 2023-5-01.

[11] Wikipedia, "2012 champions league final," https://en.wikipedia.org/wiki/2012_UEFA_Champions_League_final, 2023, accessed: 2023-4-28.

[12] ——, "Penalty box picture," https://en.wikipedia.org/wiki/Penalty_area, 2023, accessed: 2023-4-28.

[13] Xiaoyan Li, "Model training and evaluation 1," https://princeton.instructure.com/courses/10983/files/2083529?module_item_id=406847, 2023, accessed: 2023-3-23.

10. Appendix

10.1. Random Forest Classifier

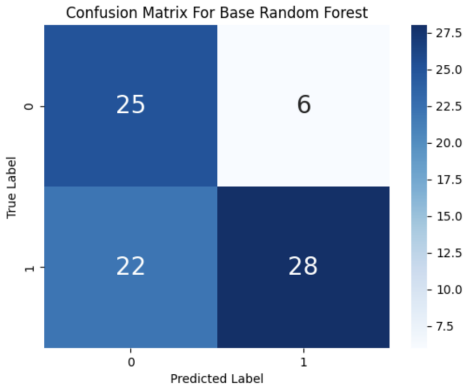


Figure 12: Confusion Matrix for Base Random Forest Model

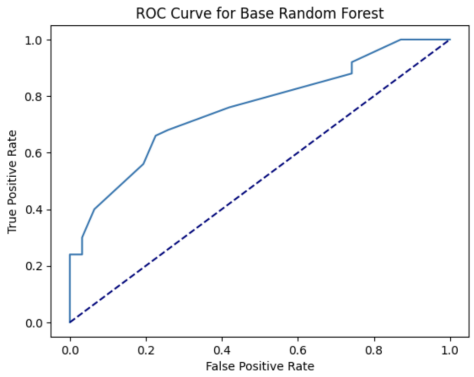


Figure 13: ROC Curve for Base Random Forest Model

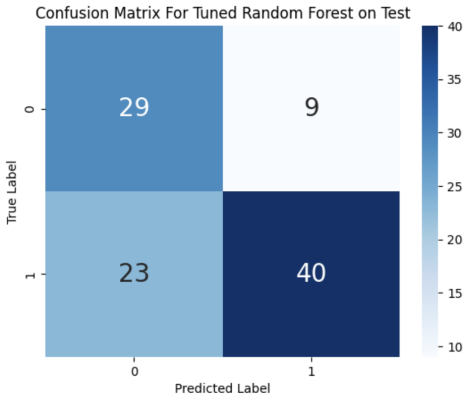


Figure 14: Confusion Matrix for Tuned Random Forest On Test Set

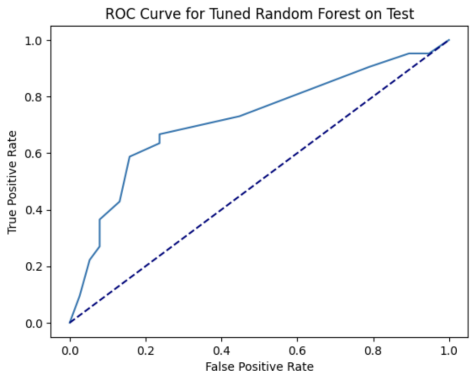


Figure 15: ROC Curve for Tuned Random Forest Model on Test Set

10.2. Decision Forest Classifier

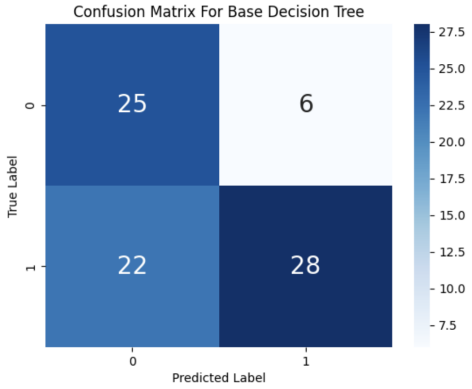


Figure 16: Confusion Matrix for Base Decision Tree Model

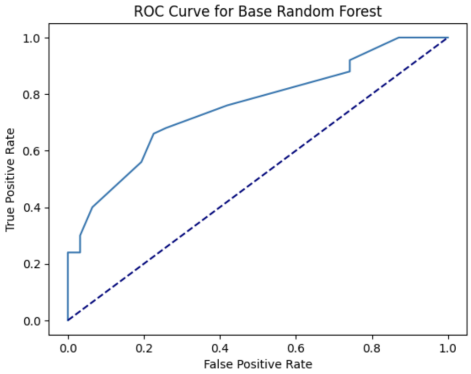


Figure 17: ROC Curve for Base Decision Tree Model

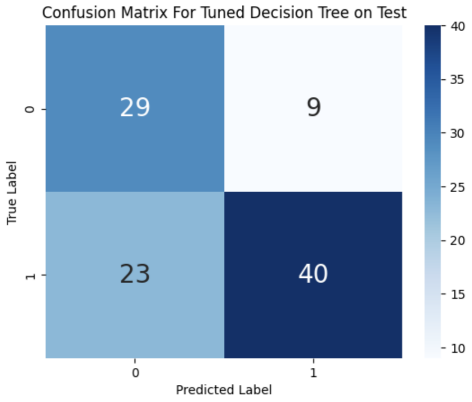


Figure 18: Confusion Matrix for Tuned Decision On Test Set

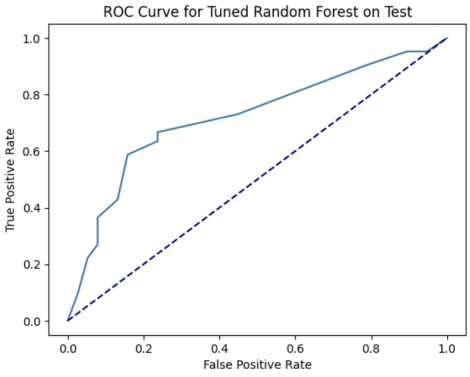


Figure 19: ROC Curve for Tuned Random Decision Tree on Test Set

10.3. Logistic Regression Classifier

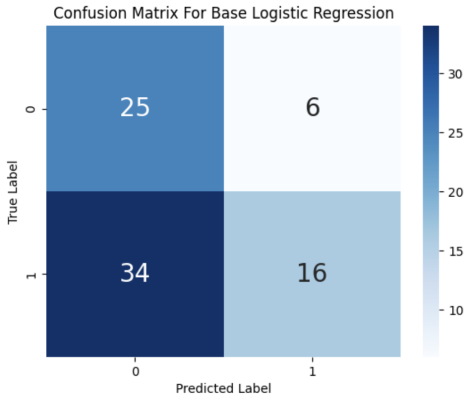


Figure 20: Confusion Matrix for Base Logistic Regression Model

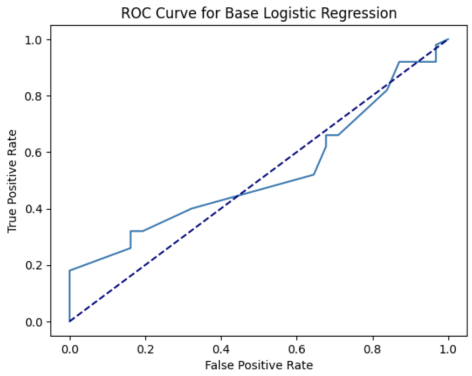


Figure 21: ROC Curve for Base Logistic Regression Model

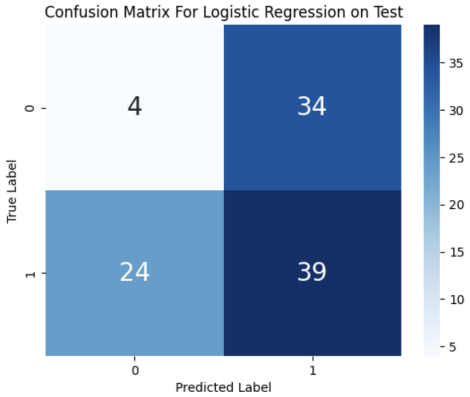


Figure 22: Confusion Matrix for Tuned Logistic Regression On Test Set

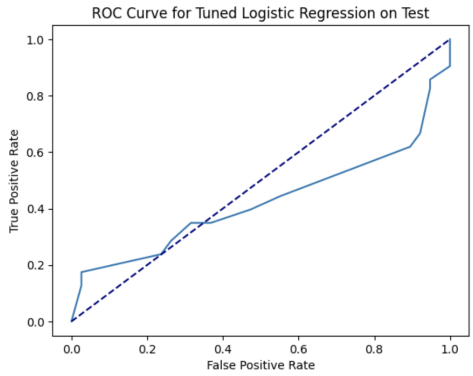


Figure 23: ROC Curve for Tuned Logistic Regression on Test Set

10.4. Support Vector Classifier

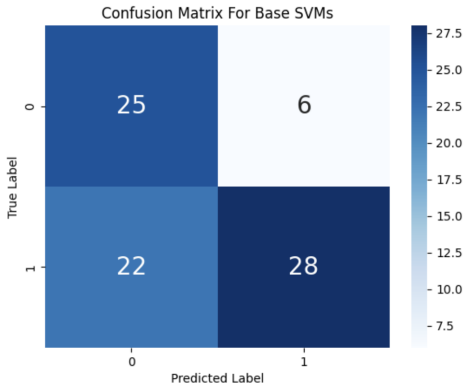


Figure 24: Confusion Matrix for Base SVC Model

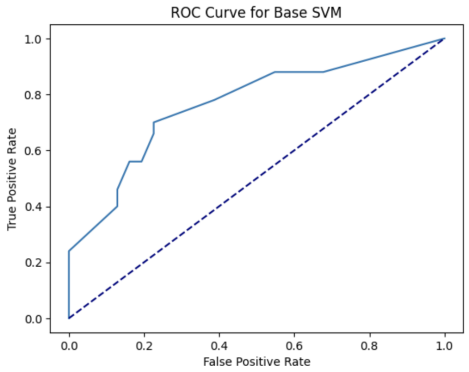


Figure 25: ROC Curve for Base SVC Model

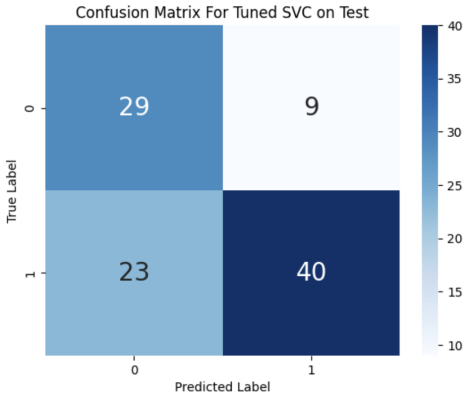


Figure 26: Confusion Matrix for Tuned SVC Model Against Test

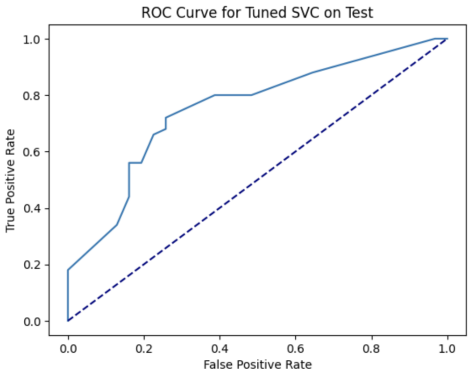


Figure 27: ROC Curve for Tuned SVC Model Against Test

10.5. Gradient Boosting

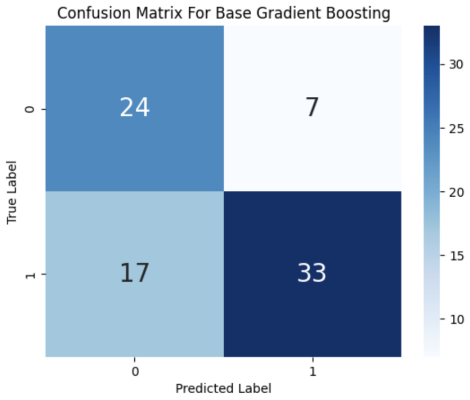


Figure 28: Confusion Matrix for Base Gradient Boosting Model

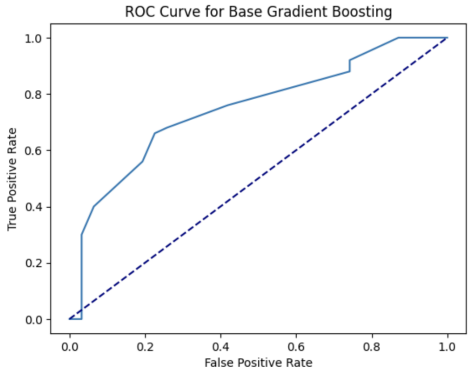


Figure 29: ROC Curve for Base Gradient Boosting Model

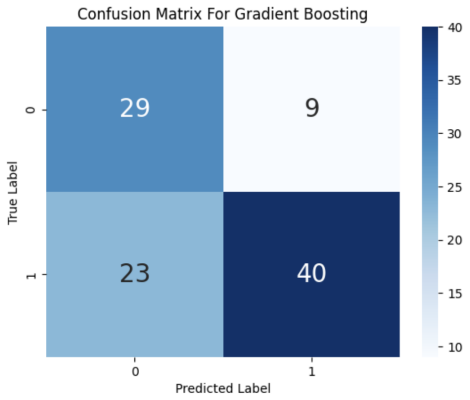


Figure 30: Confusion Matrix for Tuned Gradient Boosting On Test Set

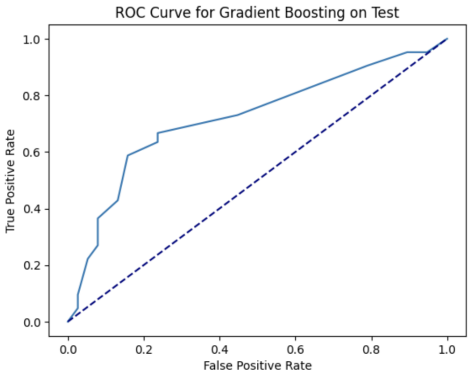


Figure 31: ROC Curve for Tuned Gradient Boosting on Test Set

AFTER ADDING THE NEW FEATURES

10.6. Random Forest Classifier

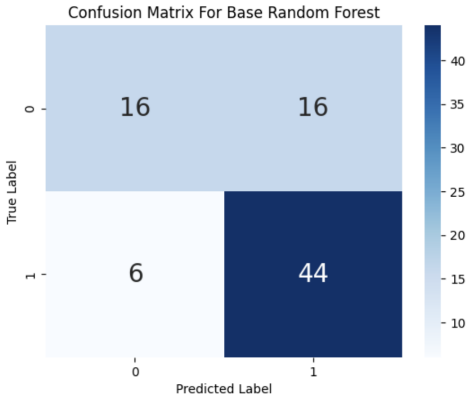


Figure 32: Confusion Matrix for Base Random Forest Model After New Features

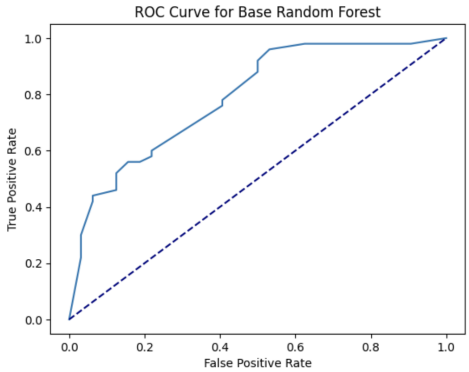


Figure 33: ROC Curve for Base Random Forest Model After New Features

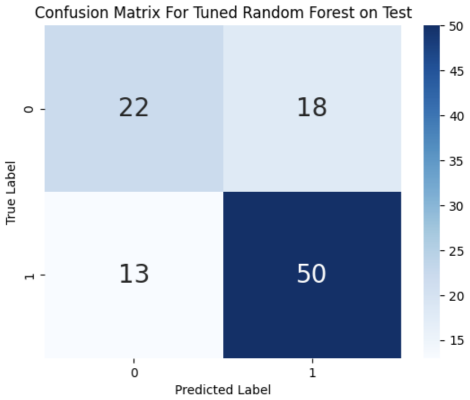


Figure 34: Confusion Matrix for Tuned Random Forest On Test Set After New Features

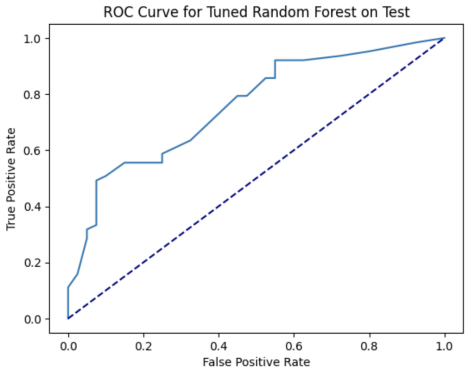


Figure 35: ROC Curve for Tuned Random Forest Model on Test Set After New Features

10.7. Decision Forest Classifier

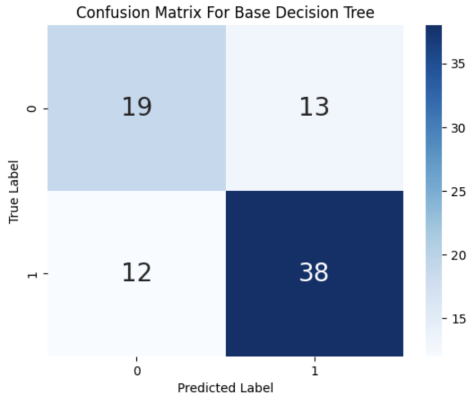


Figure 36: Confusion Matrix for Base Decision Tree Model After New Features

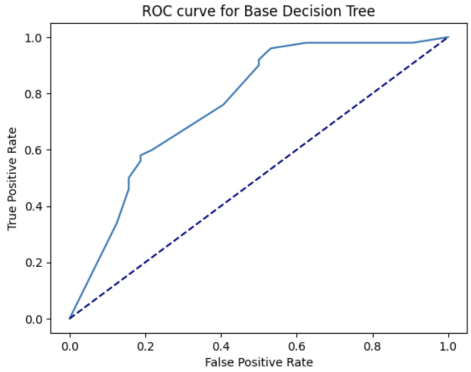


Figure 37: ROC Curve for Base Decision Tree Model After New Features

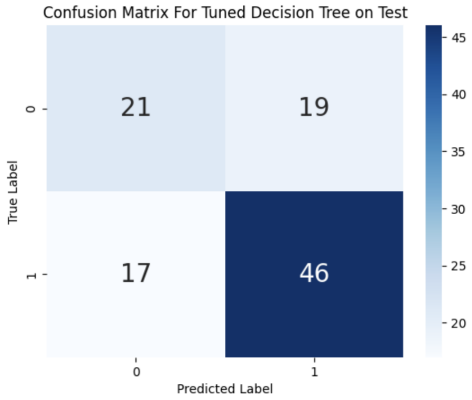


Figure 38: Confusion Matrix for Tuned Decision On Test Set After New Features

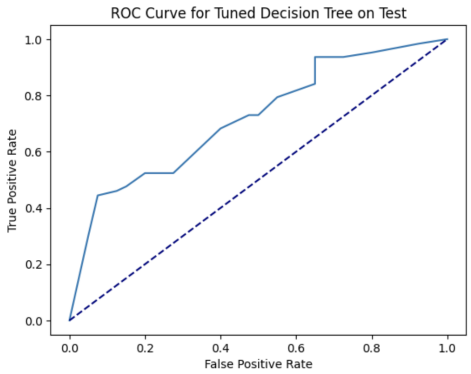


Figure 39: ROC Curve for Tuned Random Decision Tree on Test Set After New Features

10.8. Logistic Regression Classifier

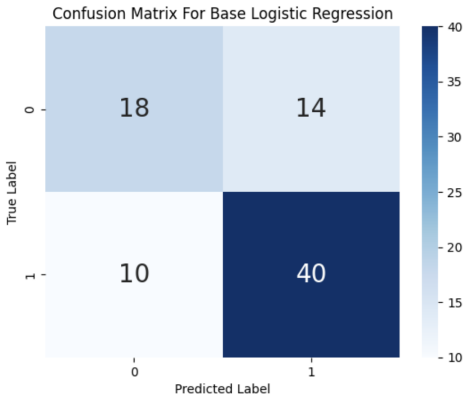


Figure 40: Confusion Matrix for Base Logistic Regression Model After New Features

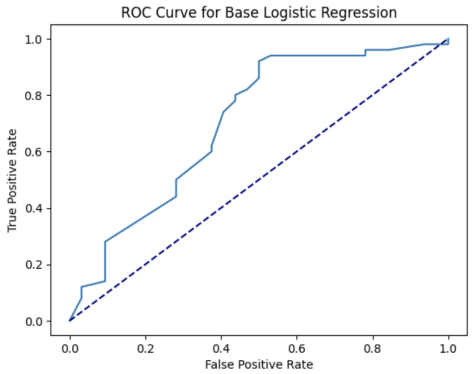


Figure 41: ROC Curve for Base Logistic Regression Model After New Features

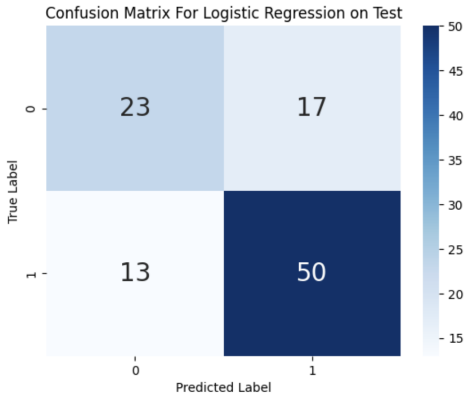


Figure 42: Confusion Matrix for Tuned Logistic Regression On Test Set After New Features

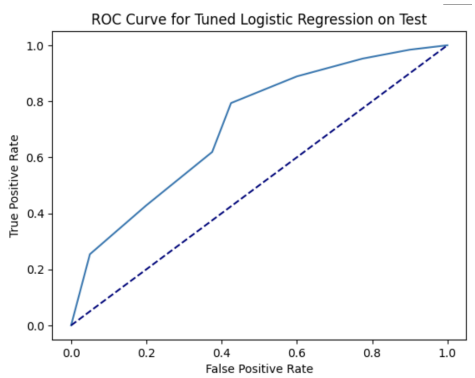


Figure 43: ROC Curve for Tuned Logistic Regression on Test Set After New Features

10.9. Support Vector Classifier

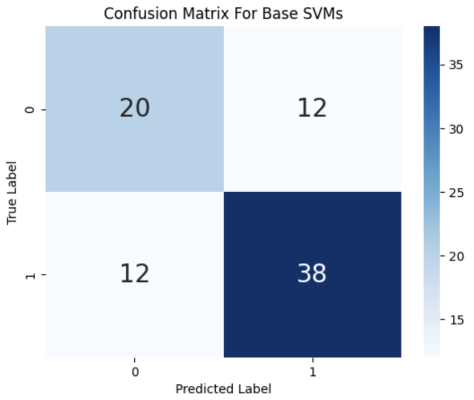


Figure 44: Confusion Matrix for Base SVC Model After New Features

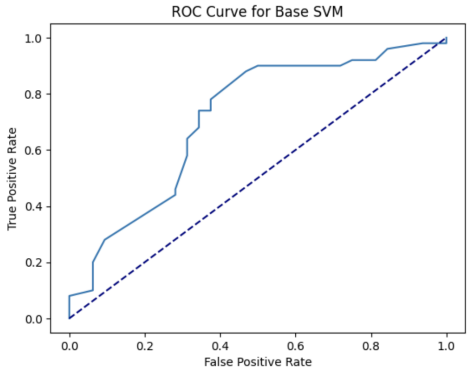


Figure 45: ROC Curve for Base SVC Model After New Features

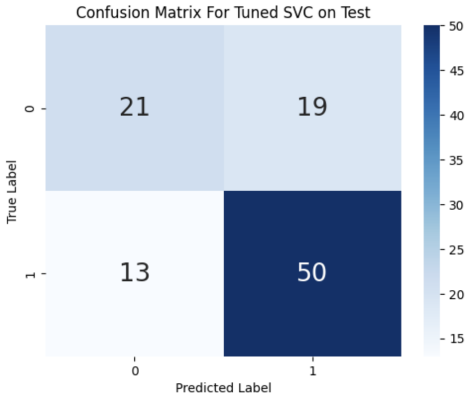


Figure 46: Confusion Matrix for Tuned SVC Model Against Test After New Features

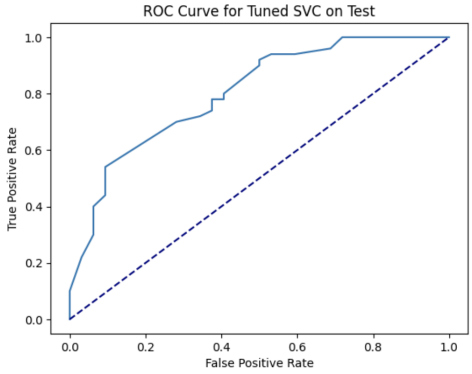


Figure 47: ROC Curve for Tuned SVC Model Against Test After New Features

10.10. Gradient Boosting

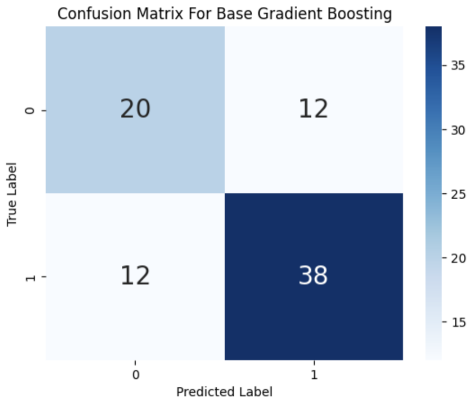


Figure 48: Confusion Matrix for Base Gradient Boosting Model After New Features

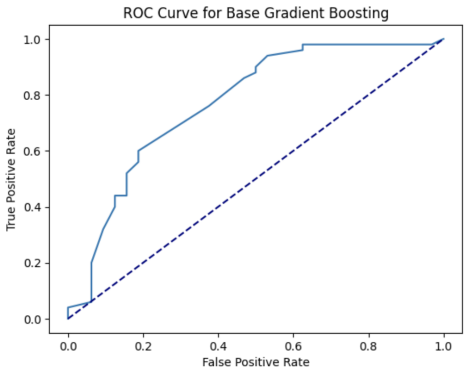


Figure 49: ROC Curve for Base Gradient Boosting Model After New Features

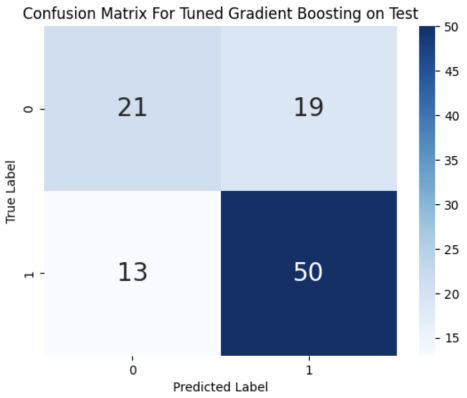


Figure 50: Confusion Matrix for Tuned Gradient Boosting On Test Set After New Features

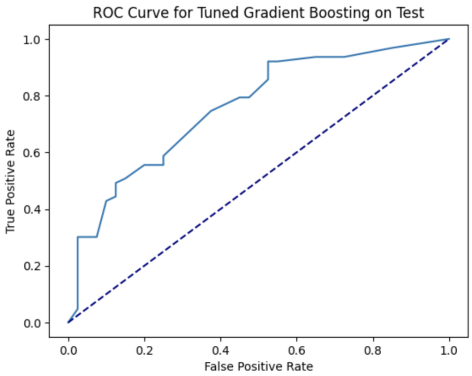


Figure 51: ROC Curve for Tuned Gradient Boosting on Test Set After New Features