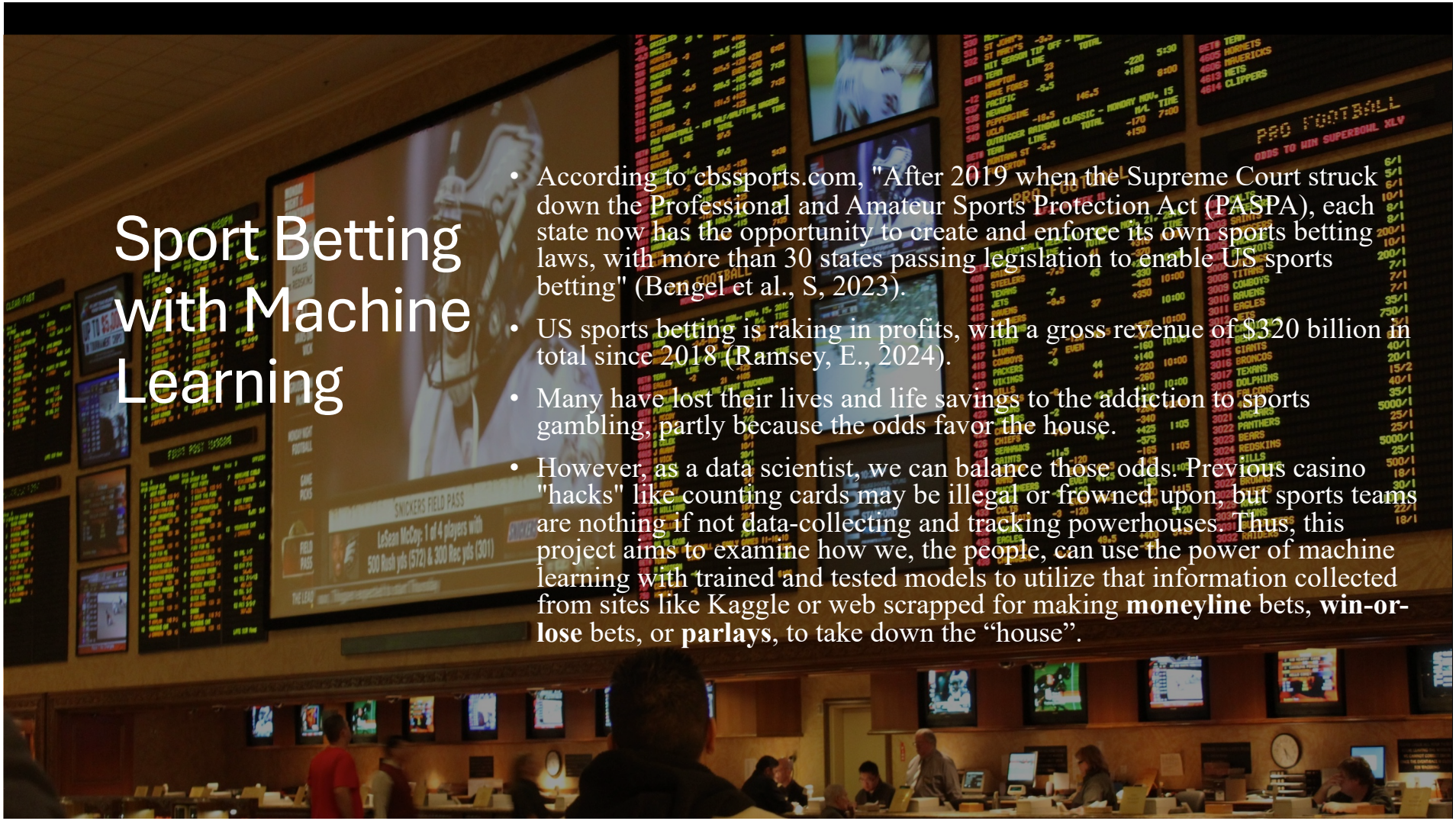
Even the Odds

By Jerimey Simons

Sport Betting with Machine Learning

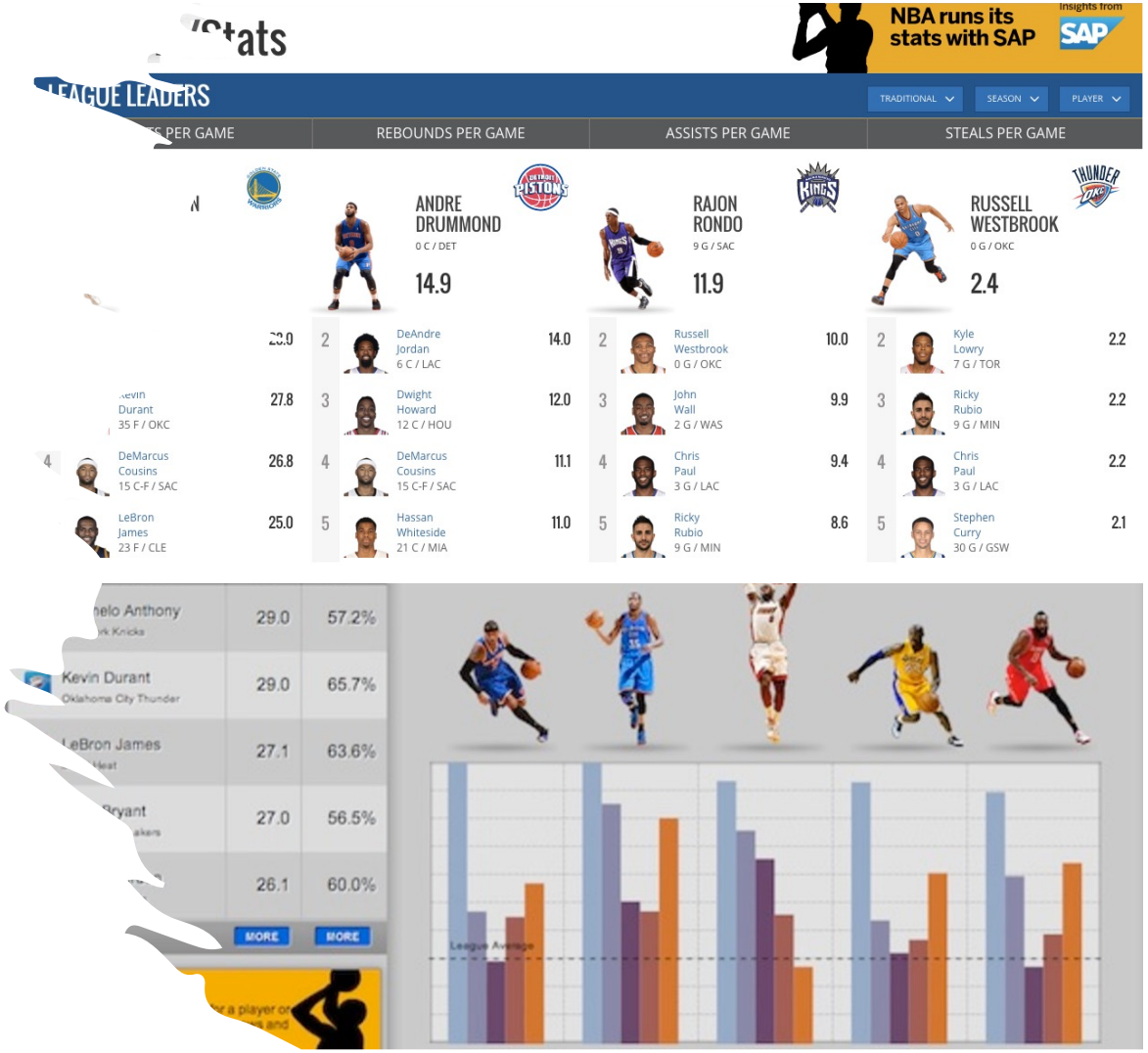


- According to cbssports.com, "After 2019 when the Supreme Court struck down the Professional and Amateur Sports Protection Act (PASPA), each state now has the opportunity to create and enforce its own sports betting laws, with more than 30 states passing legislation to enable US sports betting" (Bengel et al., S, 2023).
- US sports betting is raking in profits, with a gross revenue of $320 billion in total since 2018 (Ramsey, E., 2024).
- Many have lost their lives and life savings to the addiction to sports gambling, partly because the odds favor the house.
- However, as a data scientist, we can balance those odds. Previous casino "hacks" like counting cards may be illegal or frowned upon, but sports teams are nothing if not data-collecting and tracking powerhouses. Thus, this project aims to examine how we, the people, can use the power of machine learning with trained and tested models to utilize that information collected from sites like Kaggle or web scrapped for making **moneyline** bets, **win-or-lose** bets, or **parlays**, to take down the “house”.

NBA GAMES DATASET

NBA GAMES DATA

Description: Data collected from the NBA stats website consisting of NBA games from 2004 to 2020 into 5 datasets games, games_details, players, ranking, and teams.



```
pred_y = clf.predict(test_x)

#view prediction
print(f"Predicted outcome: {pred_y}")

#view actual targets
print(f"confirm outcome:  {test_y}")

print("Testing")
from sklearn.metrics import accuracy_score, f
print("f1:" + str(f1_score(pred_y, test_y, av
print("accuracy:" + str(accuracy_score(pred_y
print("precision:" + str(precision_score(pred
print("recall:" + str(recall_score(pred_y, te

print(" ")

train_pred_y = clf.predict(train_x)
print("trianing")
print("f1:" + str(f1_score(train_pred_y, trai
print("accuracy:" + str(accuracy_score(train_
print("precision:" + str(precision_score(train_pred_y, train_y, average='micro')))
print("recall:" + str(recall_score(train_pred_y, train_y, average='micro')))
```

Support Vector Classifier

TRAINING ACCURACY: 85%

TESTING ACCURACY: 84%

- PRESISION: 84%
- RECALL: 84%
- F1-SCORE: 84%

The model performs relatively well on both the training and testing data. It achieves an F1 score, accuracy, precision, and recall of approximately 0.841 on the testing set, indicating good performance in predicting outcome for new, unseen data.

```
pred_y = clf.predict(test_x)

#view prediction
print(f" Predicted outcome: {pred_y}")

#view actual targets
print(f" confirm outcome: {test_y}")

print("Testing")
from sklearn.metrics import accuracy_score
print("f1:" + str(f1_score(pred_y, test_y,
print("accuracy:" + str(accuracy_score(pre
print("precision:" + str(precision_score(p
print("recall:" + str(recall_score(pred_y,

print(" ")

train_pred_y = clf.predict(train_x)
print("trianing")
print("f1:" + str(f1_score(train_pred_y, t
print("accuracy:" + str(accuracy_score(tra
print("precision:" + str(precision_score(train_pred_y, train_y, average='micro')))
print("recall:" + str(recall_score(train_pred_y, train_y, average='micro')))
```

score

Decision Tree Classifier

TRAINING ACCURACY: 100%

TESTING ACCURACY: 76%

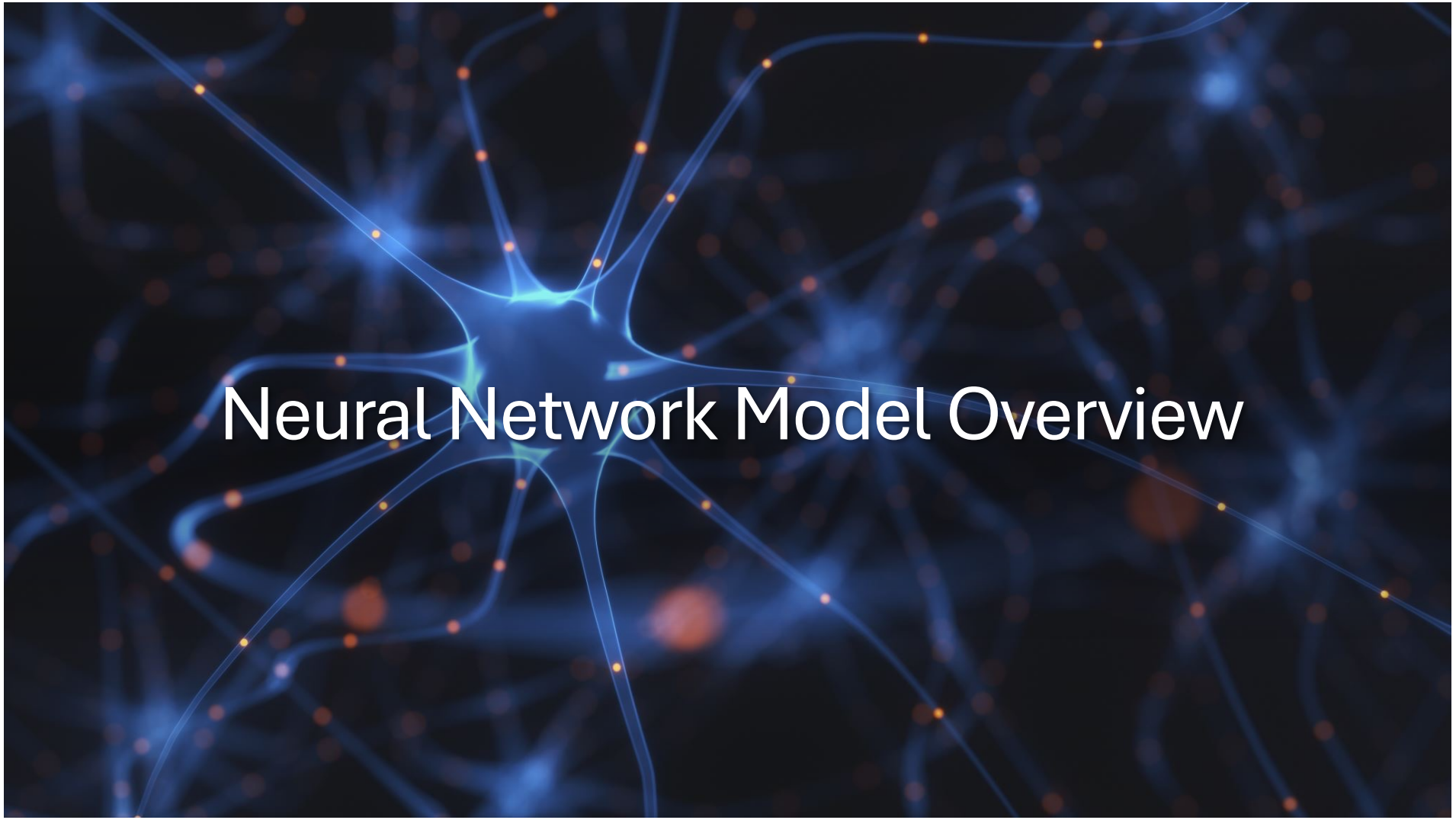
- PRESISION: 76%
- RECALL: 76%
- F1-SCORE: 76%

This indicates that the model performs perfectly on the training data, achieving perfect scores across all metrics. However, on the testing data, the model's performance suggests overfitting.

```
knn.fit(train_x, train_y)

pred_y = knn.predict(test_x)

#view prediction
print(f" Predicted outcome: {pred_y}")

#view actual targets
print(f" confirm outcome: {test_y}")

print("Testing")
from sklearn.metrics import accuracy_score,                                    core
print("f1:" + str(f1_score(pred_y, test_y,
print("accuracy:" + str(accuracy_score(pred
print("precision:" + str(precision_score(pr
print("recall:" + str(recall_score(pred_y,

#Check for over/underfitting
train_pred_y = knn.predict(train_x)
print("Training")
print("f1:" + str(f1_score(train_pred_y, tr
print("accuracy:" + str(accuracy_score(trai
print("precision:" + str(precision_score(train_pred_y, train_y, average='micro')))
print("recall:" + str(recall_score(train_pred_y, train_y, average='micro')))
```

- **Kneighbors Classifier**

- TRAINING ACCURACY: 86%
- TESTING ACCURACY: 81%
  - PRESISION: 81%
  - RECALL: 81%
  - F1-SCORE: 81%
- The results indicate the model's performance on the testing set and potential overfitting or underfitting based on the training set.


Neural Network Model Overview

Dataset At First Glance:

Dataset Size:	The dataset contains 26,652 entries with 10 columns.
Missing Values:	There are no missing values in the dataset.
Data Types:	The dataset consists mostly of float64 features, two int64 columns (id and Class), and no categorical variables.
Target Distribution:	The target variable ‘HOME_TEAM_WINS' is binary (0 and 1). 0 being a loss and 1 being a wind.
Duplicates:	No duplicates were found in the dataset.

	FG_PCT_home	FT_PCT_home	FG3_PCT_home	AST_home	REB_home	FG_PCT_away	FT_PCT_away	FG3_PCT_away	AST_away	REB_away
0	0.409	0.929	0.308	32.0	56.0	0.372	0.737	0.375	22.0	31.0
1	0.446	0.611	0.400	30.0	58.0	0.403	0.818	0.381	20.0	36.0
2	0.470	0.800	0.333	25.0	38.0	0.488	0.724	0.385	20.0	44.0
3	0.389	0.947	0.238	26.0	54.0	0.395	0.895	0.364	20.0	34.0
4	0.466	0.792	0.500	29.0	42.0	0.430	0.750	0.450	15.0	37.0

	FG_PCT_home	FT_PCT_home	FG3_PCT_home	AST_home	REB_home	FG_PCT_away	FT_PCT_away	FG3_PCT_away	AST_away	REB_away
0	-0.920366	1.671171	-0.436498	1.764291	1.904183	-1.407475	-0.213041	0.231171	0.093222	-1.702350
1	-0.266454	-1.484651	0.395086	1.379307	2.205811	-0.848741	0.570365	0.286227	-0.293990	-0.937178
2	0.157705	0.390979	-0.210524	0.416846	-0.810472	0.683271	-0.338772	0.322931	-0.293990	0.287097
3	-1.273832	1.849803	-1.069225	0.609338	1.602555	-0.992931	1.315083	0.130235	-0.293990	-1.243247
4	0.087012	0.311588	1.298982	1.186815	-0.207216	-0.362102	-0.087309	0.919373	-1.262022	-0.784144

Data Preprocessing:

1. Scaling:

1. All Selected features were scaled using StandardScaler.

2. Train-Test Split:

1. The dataset has been split into training and testing sets with a ratio of 70:30.

Model Overview

1. **Neural Network Architecture:**
    1. The neural network consists of one hidden layer with 10 units, ReLU activation, dropout regularization (dropout rate of 0.5), and an output layer with 1 unit and sigmoid activation for binary classification.
2. **Model Compilation:**
    1. Binary crossentropy is used as the loss function, and the Adam optimizer is employed.
3. **Model Training:**
    1. The model is trained for 50 epochs with a batch size of 10.

```
rom keras.models import Sequential
om keras import layers


 = Sequential()
dd(layers.Dense(10, activation='relu', input_shape = (29,)))
d(layers.Dropout(0.5))
(layers.Dense(1, activation='sigmoid'))

ile(loss='binary_crossentropy',
    optimizer='adam',
     metrics=['accuracy'])

ry()
```

```
quential_1"
_________________________________________________________________
pe)                 Output Shape              Param #
=================================================================
(Dense)             (None, 10)                110

_1 (Dropout)        (None, 10)                0

_3 (Dense)          (None, 1)                 11

=================================================================
 params: 121 (484.00 Byte)
inable params: 121 (484.00 Byte)
n-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

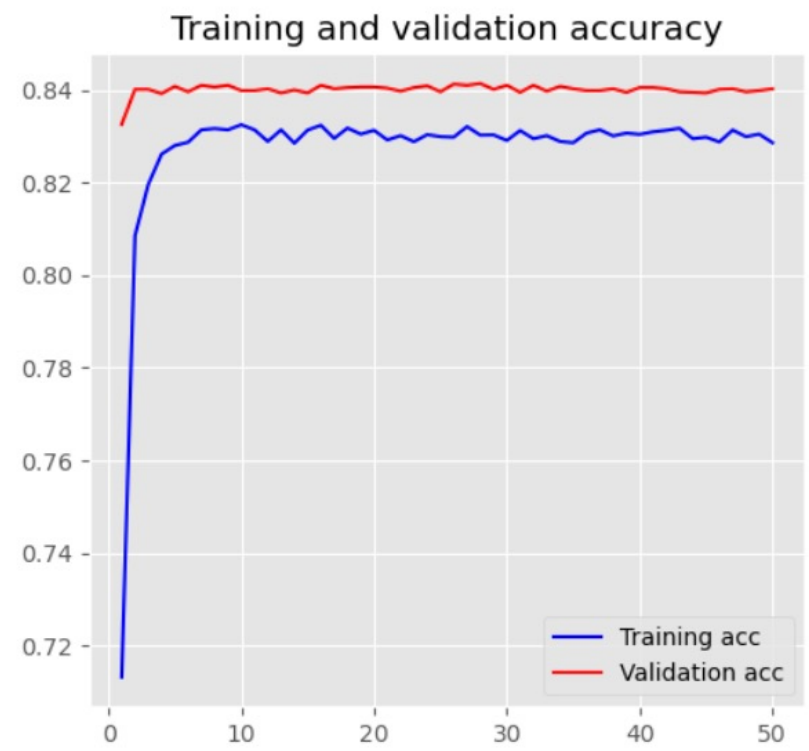
Model Evaluation

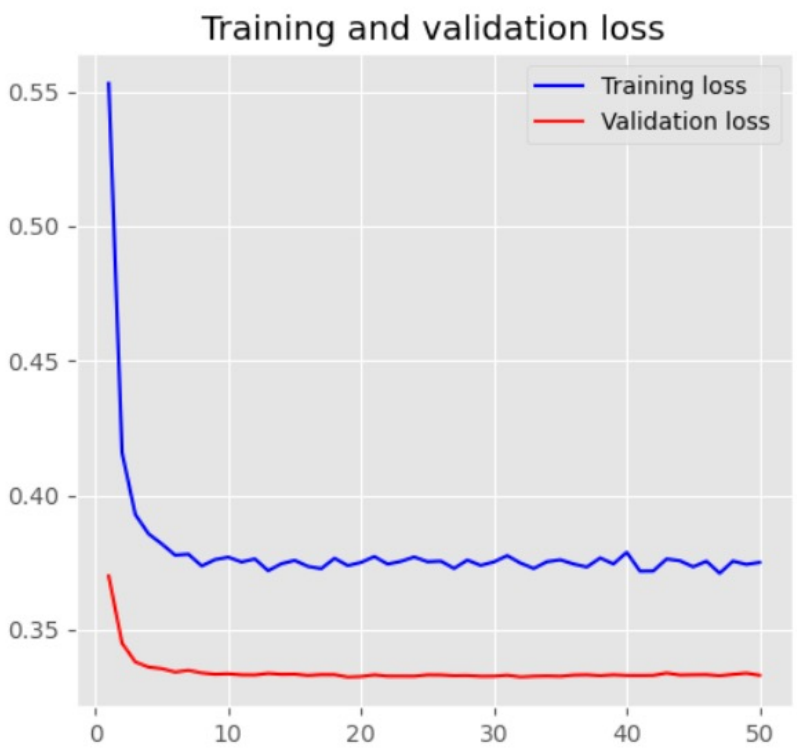
Training and Testing Accuracy:

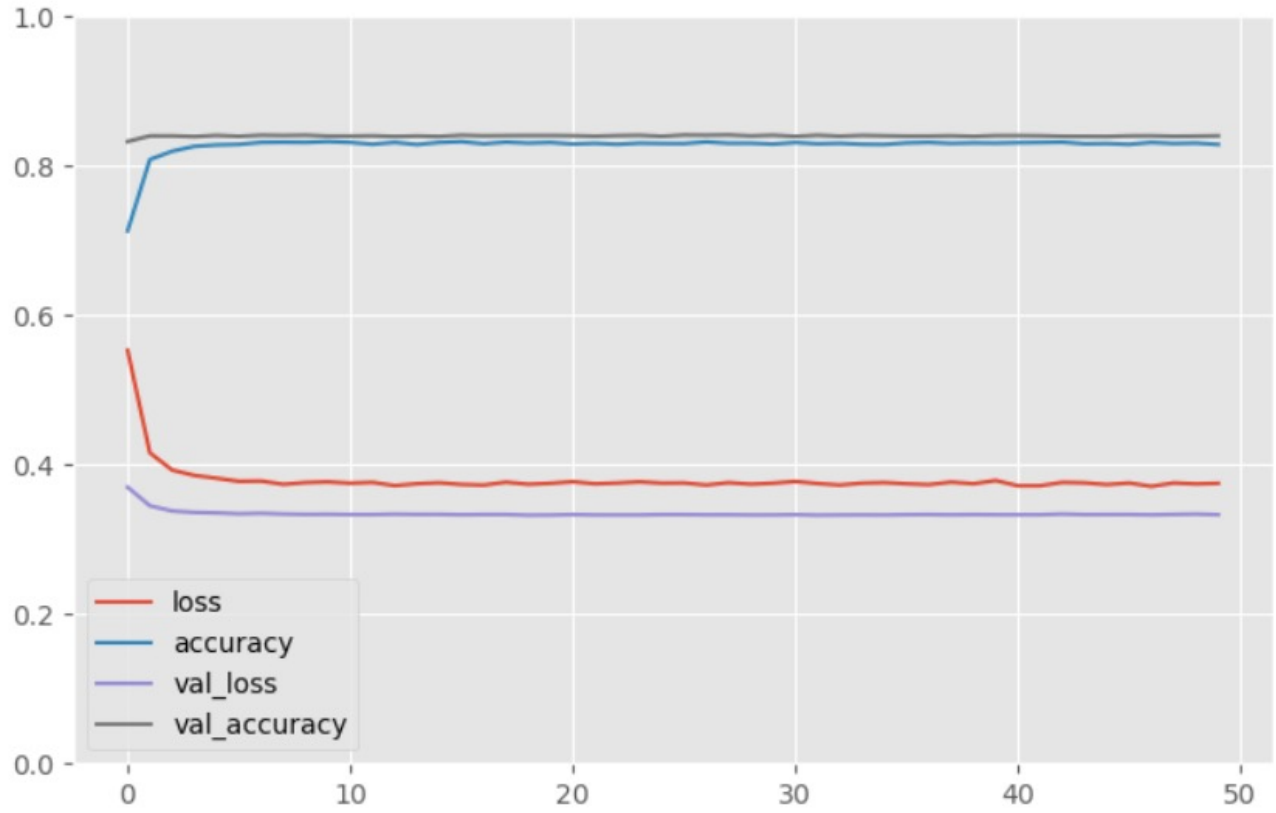
- The model achieved an accuracy of approximately 84.09% on the training set and 84.03% on the testing set.

Testing Precision, Recall, and F1-score:

- Precision: 84.14%
- Recall: 89.77%
- F1-score: 86.87%


Training and validation accuracy
0.84
0.82
0.80
0.78
0.76
0.74
0.72
0
10
20
30
40
50
Training acc
Validation acc


Training and validation loss
0.55
0.50
0.45
0.40
0.35
0
10
20
30
40
50
Training loss
Validation loss


1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
loss
accuracy
val_loss
val_accuracy

Conclusion

The neural network model shows promising performance in binary classification tasks, achieving satisfactory accuracy and demonstrating good generalization to unseen data.

Further optimization and fine-tuning may enhance the model's performance.