Documentation and screenshots

Screenshots:

- EDA performed on the datasets

  - ```
    data.info()
    ```

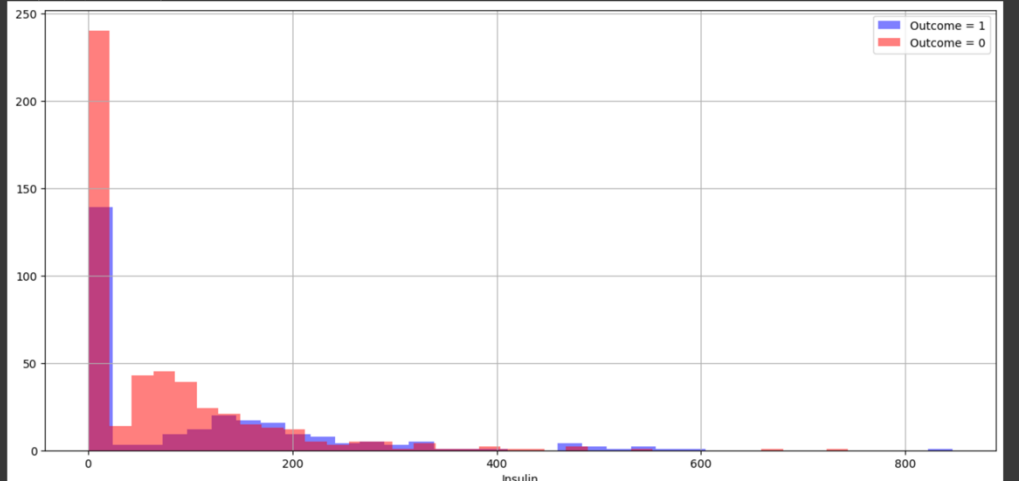
    ```
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 768 entries, 0 to 767
    Data columns (total 9 columns):
     #   Column                    Non-Null Count  Dtype
    ---  ------                    --------------  -----
     0   Pregnancies               768 non-null    int64
     1   Glucose                   768 non-null    int64
     2   BloodPressure             768 non-null    int64
     3   SkinThickness             768 non-null    int64
     4   Insulin                   768 non-null    int64
     5   BMI                       768 non-null    float64
     6   DiabetesPedigreeFunction  768 non-null    float64
     7   Age                       768 non-null    int64
     8   Outcome                   768 non-null    int64
    dtypes: float64(2), int64(7)
    memory usage: 54.1 KB
    ```

    ```
    [195] data.describe()
    ```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

  - ```
    plt.figure(figsize=(15, 7))
    data[data['Outcome']==1]['Insulin'].hist(alpha=0.5, color='blue', bins= 35, label = 'Outcome = 1')
    data[data['Outcome']==0]['Insulin'].hist(alpha=0.5, color='red', bins= 35, label = 'Outcome = 0')
    plt.legend()
    plt.xlabel('Insulin')
    ```

    ```
    Text(0.5, 0, 'Insulin')
    ```

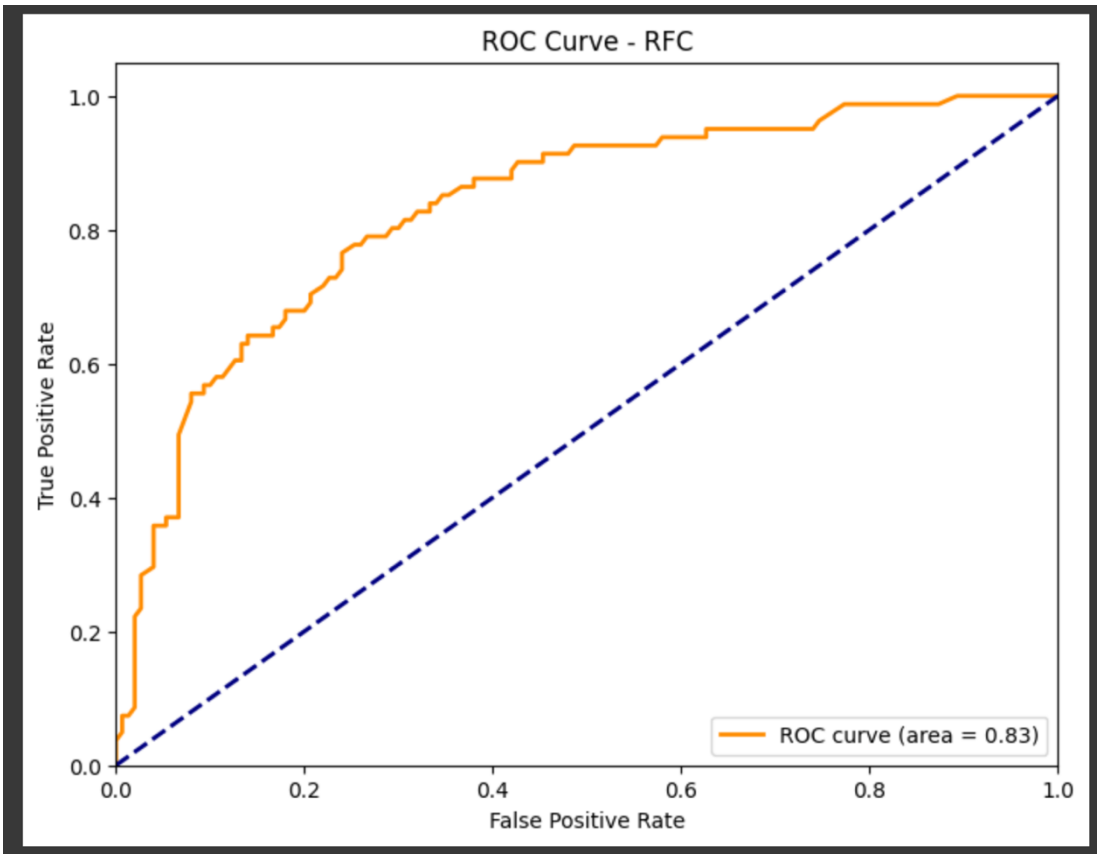
- Random Forest Classification metrics and overall metric, ROC curve
  - metrics_RFC

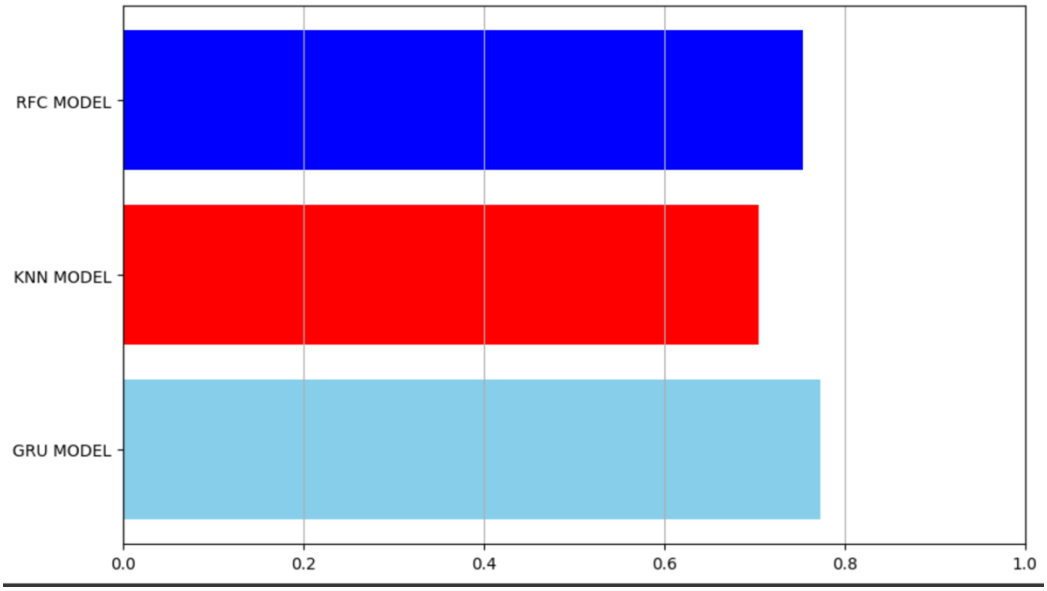
	Fold	True Positives (TP)	True Negatives (TN)	False Positives (FP)	False Negatives (FN)	True Positive Rate (TPR)	True Negative Rate (TNR)	False Positive Rate (FPR)	False Negative Rate (FNR)	Precision	F1Measure	Accuracy	Recall	ErrorRate	BACC	TSS
0	1	9	29	8	8	0.529412	0.783784	0.216216	0.470588	0.529412	0.529412	0.703704	0.529412	0.296296	0.656598	0.313196
1	2	9	31	8	6	0.600000	0.794872	0.205128	0.400000	0.529412	0.562500	0.740741	0.600000	0.259259	0.697436	0.394872
2	3	9	28	7	10	0.473684	0.800000	0.200000	0.526316	0.562500	0.514286	0.685185	0.473684	0.314815	0.636842	0.273684
3	4	13	29	2	10	0.565217	0.935484	0.064516	0.434783	0.866667	0.684211	0.777778	0.565217	0.222222	0.750351	0.500701
4	5	15	29	4	6	0.714286	0.878788	0.121212	0.285714	0.789474	0.750000	0.814815	0.714286	0.185185	0.796537	0.593074
5	6	9	37	1	7	0.562500	0.973684	0.026316	0.437500	0.900000	0.692308	0.851852	0.562500	0.148148	0.768092	0.536184
6	7	12	30	5	7	0.631579	0.857143	0.142857	0.368421	0.705882	0.666667	0.777778	0.631579	0.222222	0.744361	0.488722
7	8	12	25	4	12	0.500000	0.862069	0.137931	0.500000	0.750000	0.600000	0.698113	0.500000	0.301887	0.681034	0.362069
8	9	8	28	7	10	0.444444	0.800000	0.200000	0.555556	0.533333	0.484848	0.679245	0.444444	0.320755	0.622222	0.244444
9	10	8	35	3	7	0.533333	0.921053	0.078947	0.466667	0.727273	0.615385	0.811321	0.533333	0.188679	0.727193	0.454386

Next steps: View recommended plots

[210] overall_metrics_df_RFC

	overall_tp	overall_tn	overall_fp	overall_fn	overall_tpr	overall_tnr	overall_fpr	overall_fnr	overall_precision	overall_f1_measure	overall_accuracy
0	10.4	30.1	4.9	8.3	0.555446	0.860688	0.139312	0.444554	0.689395	0.609962	0.754053

  - 

- Accuracy model comparison
  - 

- Metrics for each model side by side
  - Metrics for Each Model:

Fold	True Positives (TP) GRU	TP KNN	TP RFC	True Negatives (TN) GRU	TN KNN	TN RFC
0 1	10	8	9	33	32	29
1 2	11	11	9	32	32	31
2 3	14	14	9	29	29	28
3 4	12	11	13	27	27	29
4 5	6	4	15	32	31	29
5 6	16	11	9	27	26	37
6 7	4	7	12	35	34	30
7 8	10	6	12	35	32	25
8 9	10	6	8	32	23	28
9 10	13	11	8	27	24	35

Fold	False Positives (FP) GRU	FP KNN	FP RFC	False Negatives (FN) GRU	...	ErrorRate RFC
0 1	5	6	8	6	...	0.296296
1 2	5	5	8	6	...	0.259259
2 3	3	3	7	8	...	0.314815
3 4	9	9	2	6	...	0.222222
4 5	5	6	4	11	...	0.185185
5 6	6	7	1	5	...	0.148148
6 7	1	2	5	14	...	0.222222
7 8	4	7	4	4	...	0.301887
8 9	3	12	7	8	...	0.320755
9 10	1	4	3	12	...	0.188679

Fold	BACC GRU	BACC KNN	BACC RFC	TSS GRU	TSS KNN	TSS RFC	HSS GRU
0 1	0.746711	0.671053	0.656598	0.493421	0.342105	0.313196	0.001687
1 2	0.755962	0.755962	0.697436	0.511924	0.511924	0.394872	0.001684
2 3	0.771307	0.771307	0.636842	0.542614	0.542614	0.273684	0.001725
3 4	0.708333	0.680556	0.750351	0.416667	0.361111	0.500701	0.001203
4 5	0.608903	0.536566	0.796537	0.217806	0.073132	0.593074	0.000921
5 6	0.790043	0.655844	0.768092	0.580087	0.311688	0.536184	0.001648
6 7	0.597222	0.666667	0.744361	0.194444	0.333333	0.488722	0.001587
7 8	0.805861	0.624542	0.681034	0.611722	0.249084	0.362069	0.002241
8 9	0.734921	0.495238	0.622222	0.469841	-0.009524	0.244444	0.001807
9 10	0.742143	0.648571	0.727193	0.484286	0.297143	0.454386	0.001774

Steps to run the program

- Installation Instructions
    - Python is required
    - Required Python packages: pandas, numpy, matplotlib, seaborn, scikit-learn, keras, tensorflow

- Installation steps
    - On terminal download Python packages: pandas, numpy, matplotlib, seaborn, scikit-learn, keras, tensorflow
    - They can be installed using pip
        - Ex: pip install pandas numpy matplotlib seaborn scikit-learn keras tensorflow

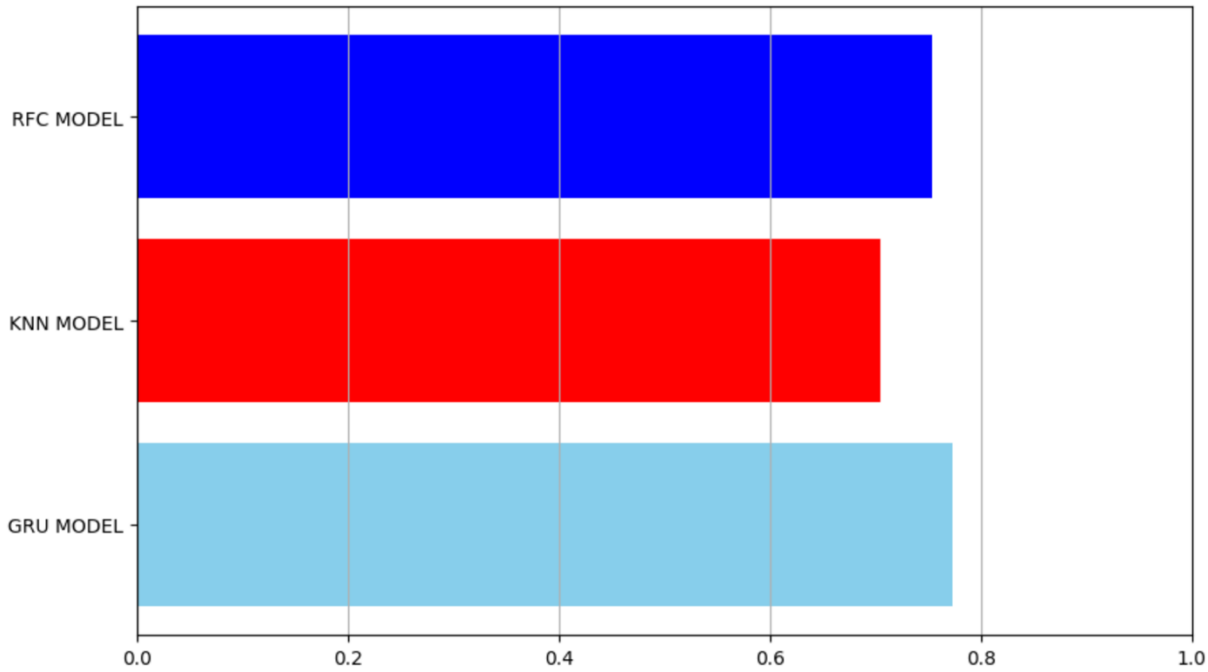
- Running the program
    - Navigate to where the program files are
    - Execute the main Python script FinalProject_Python.py.py

- Data requirements
    - The program requires a dataset in CSV format. To download the dataset used for this program use the following link: **https://www.kaggle.com/datasets/akshaydattatraykhare/diabetes-dataset**
    - Ensure the dataset is stored in the directory within the project folder.

- Expected output
    - After running the program, the program will generate predictions for the diabetes dataset using three different machine learning models. These models are: **Random forest, KNN, and GRU.**
    - Output includes performance metrics such as accuracy, precision, recall, F1-score, and ROC curves for each model.

Experimental results

- Accuracy comparison between the three models



- Random Forest
  - Classification performance

	Fold	True Positives (TP)	True Negatives (TN)	False Positives (FP)	False Negatives (FN)	True Positive Rate (TPR)	True Negative Rate (TNR)	False Positive Rate (FPR)	False Negative Rate (FNR)	Precision	F1Measure	Accuracy	Recall	ErrorRate	BACC	TSS	HSS
0	1	9	29	8	8	0.529412	0.783784	0.216216	0.470588	0.529412	0.529412	0.703704	0.529412	0.296296	0.656598	0.313196	0.000996
1	2	9	31	8	6	0.600000	0.794872	0.205128	0.400000	0.529412	0.562500	0.740741	0.600000	0.259259	0.697436	0.394872	0.001256
2	3	9	28	7	10	0.473684	0.800000	0.200000	0.526316	0.562500	0.514286	0.685185	0.473684	0.314815	0.636842	0.273684	0.000900
3	4	13	29	2	10	0.565217	0.935484	0.064516	0.434783	0.866667	0.684211	0.777778	0.565217	0.222222	0.750351	0.500701	0.001712
4	5	15	29	4	6	0.714286	0.878788	0.121212	0.285714	0.789474	0.750000	0.814815	0.714286	0.185185	0.796537	0.593074	0.001784
5	6	9	37	1	7	0.562500	0.973684	0.026316	0.437500	0.900000	0.692308	0.851852	0.562500	0.148148	0.768092	0.536184	0.002437
6	7	12	30	5	7	0.631579	0.857143	0.142857	0.368421	0.705882	0.666667	0.777778	0.631579	0.222222	0.744361	0.488722	0.001554
7	8	12	25	4	12	0.500000	0.862069	0.137931	0.500000	0.750000	0.600000	0.698113	0.500000	0.301887	0.681034	0.362069	0.001223
8	9	8	28	7	10	0.444444	0.800000	0.200000	0.555556	0.533333	0.484848	0.679245	0.444444	0.320755	0.622222	0.244444	0.000858
9	10	8	35	3	7	0.533333	0.921053	0.078947	0.466667	0.727273	0.615385	0.811321	0.533333	0.188679	0.727193	0.454386	0.001967

  - Overall classification performance

	overall_tp	overall_tn	overall_fp	overall_fn	overall_tpr	overall_tnr	overall_fpr	overall_fnr	overall_precision	overall_f1_measure	overall_accuracy
0	10.4	30.1	4.9	8.3	0.555446	0.860688	0.139312	0.444554	0.689395	0.609962	0.754053

- KNN
    - Classification performance

Fold	True Positives (TP)	True Negatives (TN)	False Positives (FP)	False Negatives (FN)	True Positive Rate (TPR)	True Negative Rate (TNR)	False Positive Rate (FPR)	False Negative Rate (FNR)	Precision	F1Measure	Accuracy	Recall	ErrorRate	BACC	TSS	HSS
1	8	32	6	8	0.500000	0.842105	0.157895	0.500000	0.571429	0.533333	0.740741	0.500000	0.259259	0.671053	0.342105	0.001222
2	11	32	5	6	0.647059	0.864865	0.135135	0.352941	0.687500	0.666667	0.796296	0.647059	0.203704	0.755962	0.511924	0.001684
3	14	29	3	8	0.636364	0.906250	0.093750	0.363636	0.823529	0.717949	0.796296	0.636364	0.203704	0.771307	0.542614	0.001725
4	11	27	9	7	0.611111	0.750000	0.250000	0.388889	0.550000	0.578947	0.703704	0.611111	0.296296	0.680556	0.361111	0.001062
5	4	31	6	13	0.235294	0.837838	0.162162	0.764706	0.400000	0.296296	0.648148	0.235294	0.351852	0.536566	0.073132	0.000332
6	11	26	7	10	0.523810	0.787879	0.212121	0.476190	0.611111	0.564103	0.685185	0.523810	0.314815	0.655844	0.311688	0.000962
7	7	34	2	11	0.388889	0.944444	0.055556	0.611111	0.777778	0.518519	0.759259	0.388889	0.240741	0.666667	0.333333	0.001646
8	6	32	7	8	0.428571	0.820513	0.179487	0.571429	0.461538	0.444444	0.716981	0.428571	0.283019	0.624542	0.249084	0.000958
9	6	23	12	12	0.333333	0.657143	0.342857	0.666667	0.333333	0.333333	0.547170	0.333333	0.452830	0.495238	-0.009524	-0.000030
10	11	24	4	14	0.440000	0.857143	0.142857	0.560000	0.733333	0.550000	0.660377	0.440000	0.339623	0.648571	0.297143	0.001043

    - Overall Classification performance

overall_tp	overall_tn	overall_fp	overall_fn	overall_tpr	overall_tnr	overall_fpr	overall_fnr	overall_precision	overall_f1_measure	overall_accuracy
8.9	29.0	6.1	9.7	0.474443	0.826818	0.173182	0.525557	0.594955	0.520359	0.705416

- GRU
    - Classification performance

Fold	True Positives (TP)	True Negatives (TN)	False Positives (FP)	False Negatives (FN)	True Positive Rate (TPR)	True Negative Rate (TNR)	False Positive Rate (FPR)	False Negative Rate (FNR)	Precision	F1Measure	Accuracy	Recall	ErrorRate	BACC	TSS	HSS
1	10	33	5	6	0.625000	0.868421	0.131579	0.375000	0.666667	0.645161	0.796296	0.625000	0.203704	0.746711	0.493421	0.001687
2	11	32	5	6	0.647059	0.864865	0.135135	0.352941	0.687500	0.666667	0.796296	0.647059	0.203704	0.755962	0.511924	0.001684
3	14	29	3	8	0.636364	0.906250	0.093750	0.363636	0.823529	0.717949	0.796296	0.636364	0.203704	0.771307	0.542614	0.001725
4	12	27	9	6	0.666667	0.750000	0.250000	0.333333	0.571429	0.615385	0.722222	0.666667	0.277778	0.708333	0.416667	0.001203
5	6	32	5	11	0.352941	0.864865	0.135135	0.647059	0.545455	0.428571	0.703704	0.352941	0.296296	0.608903	0.217806	0.000921
6	16	27	6	5	0.761905	0.818182	0.181818	0.238095	0.727273	0.744186	0.796296	0.761905	0.203704	0.790043	0.580087	0.001648
7	4	35	1	14	0.222222	0.972222	0.027778	0.777778	0.800000	0.347826	0.722222	0.222222	0.277778	0.597222	0.194444	0.001587
8	10	35	4	4	0.714286	0.897436	0.102564	0.285714	0.714286	0.714286	0.849057	0.714286	0.150943	0.805861	0.611722	0.002241
9	10	32	3	8	0.555556	0.914286	0.085714	0.444444	0.769231	0.645161	0.792453	0.555556	0.207547	0.734921	0.469841	0.001807
10	13	27	1	12	0.520000	0.964286	0.035714	0.480000	0.928571	0.666667	0.754717	0.520000	0.245283	0.742143	0.484286	0.001774

    - Overall classification performance

overall_tp	overall_tn	overall_fp	overall_fn	overall_tpr	overall_tnr	overall_fpr	overall_fnr	overall_precision	overall_f1_measure	overall_accuracy
10.6	30.9	4.2	8.0	0.5702	0.882081	0.117919	0.4298	0.723394	0.619186	0.772956

Conclusion

Based on the classification performances, it can be seen that the algorithm GRU had the best performance with an overall accuracy of 77.29%, followed by Random Forest with an overall accuracy of 75.40%, and finally KNN with an accuracy of 70.54%. The reason why GRU would perform better as models like this have an architecture that allows them to adapt to different sets of data as well as capture complex patterns.

GITHUB Link:

https://github.com/BSantiagoP/Data-Mining-Final-Project/blob/main/Final_Project_Data_Mining.ipynb