Project Report on OSDA

Sharmi Islam

Project Based on FCALC

Abstract

This project explores the use of Binarized Binary Classification from the fcalc package to predict outcomes for given data set. The data set comprises diverse board configurations labeled with game outcomes (positive/negative).Split into training/testing sets, the data trains the BinarizedBinaryClassifier. Model evaluation via accuracy, precision, recall, and F1-score aims to understand the predictive capacity of binarized classification in determining game outcomes.

1 Overview

This project aims to predict the outcomes of three different data set using a Binarized Binary Classifier implemented through the 'fcalc' package. The various configurations of data set's board states represented as features, with the target variable indicating whether the outcome was positive (win) or negative (loss/draw).

The data set is preprocessed and transformed into a suitable format for machine learning, utilizing Pandas for data manipulation and Scikit-learn for splitting the data into training and testing sets. The features are encoded using one-hot encoding to prepare them for the classifier.

The 'fcalc' package's 'BinarizedBinaryClassifier' is employed to train a model on the transformed data, aiming to predict outcomes based on the board configurations. Different methods available within the classifier, such as "standard-support" or other applicable strategies, are explored to determine the most effective approach.

The performance of the classifier is evaluated using the test set, considering metrics like accuracy, precision, recall, and F1-score. Insights gained from this analysis can provide valuable understanding regarding the feasibility of predicting outcomes using binarized classification methods.

2 Binarization

2.1 Dataset 1

```
column_names = [
                'age',
                'sex',
                'cp',
                'trestbps',
                'chol',
                'fbs',
                'restecg',
                'thalach',
                'exang',
                'oldpeak',
                'slope',
                'ca',
                'thal',
                'target'
]
df = pd.read_csv('../data_sets/heart.csv', names=column_names)
df['target'] = [x == '1' for x in df['target']]
df.head()
```

Figure 1: Using proposed algorithm:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	False
1	63	1	3	145	233	1	0	150	0	2.3	0	0	1	True
2	37	1	2	130	250	0	1	187	0	3.5	0	0	2	True
3	41	0	1	130	204	0	0	172	0	1.4	2	0	2	True
4	56	1	1	120	236	0	1	178	0	0.8	2	0	2	True

Figure 2: After Binarized data table

2.2 Data set 2

```
df = pd.read_csv('../data_sets/cost_of_living.csv', names=["Countries","Cost of living, 2017","Global rank","Available data"])
# print(df.columns)
df['Cost of living, 2017'] = [float(x) >= 100.0 for x in df['Cost of living, 2017']]
df.sample(10)
```

Figure 3: Using proposed algorithm:

	Countries	Cost of living, 2017	Global rank	Available data
89	Colombia	False	89.0	2017 - 2017
97	Bosnia & Herz.	False	97.0	2017 - 2017
75	Namibia	False	75.0	2017 - 2017
69	Bahrain	False	69.0	2017 - 2017
130	Bolivia	False	130.0	2017 - 2017
114	Chad	False	114.0	2017 - 2017
110	Togo	False	110.0	2017 - 2017
45	Estonia	False	45.0	2017 - 2017
115	Cameroon	False	115.0	2017 - 2017
56	Kuwait	False	56.0	2017 - 2017

Figure 4: After Binarized data table

2.3 Dataset 3

```
column_names = [
            'Animal','Height (cm)','Weight (kg)','Color','Lifespan (years)','Diet','Habitat','Predators','Average Speed (km/h)','Countries Found','Conservation Status','Family',
]
df = pd.read_csv('../data_sets/animal_dataset_2.csv')
df['Offspring per Birth'] = [x == '1' for x in df['Offspring per Birth']]
df.sample(5)
```

Figure 5: Using proposed algorithm:

	Animal	Height (cm)	Weight (kg)	Color	Lifespan (years)	Diet	Habitat	Predators	Average Speed (km/h)	Countries Found	Conservation Status	Family	Gestation Period (days)	Top Speed (km/h)	Social Structure	Offspring per Birth
192	White Tiger	240-260	100-300	White, Black Stripes	10-15	Carnivore	Rainforests	Humans, Leopards	22	India, Southeast Asia	Critically Endangered	Felidae	230-260	22	Solitary	True
60	European Hedgehog	14-30	0.6-1.2	Brown, Spiny	2-5	Omnivore	Gardens, Grasslands	Foxes, Badgers	4-6	Europe, Asia	Least Concern	Erinaceidae	35	4-6	Solitary	False
77	Goblin Shark	Up to 320	Up to 210	Pinkish-gray	30-35	Carnivore	Deep-sea	Larger Fish, Other Sharks	1.6-3	Worldwide	Data Deficient	Mitsukurinidae	Not Applicable	Not Applicable	Solitary	False
18	Aye-Aye	35-37	2.2-2.7	Black, Brown	20-23	Omnivore, Insectivore	Rainforests	Birds of Prey	32	Madagascar	Endangered	Daubentoniidae	160-170	32	Solitary	True
144	Rottweiler	56-69	35-60	Black, Tan	8-10	Carnivore	Worldwide	Wolves, Coyotes	35	Worldwide	Not Applicable	Canidae	61-65	32	Pack-based	False

+ Code + Markdown

Figure 6: Binarized data table

3 Pattern structures

```
column_names = [
            'age',
            'sex',
            'cp',
            'trestbps',
            'chol',
            'fbs',
            'restecg',
            'thalach',
            'exang',
            'oldpeak',
            'slope',
            'ca',
            'thal',
            'target'
]
df = pd.read_csv('../data_sets/heart.csv', names = column_names)
df['target'] = [x == '1' for x in df['target']]
df.head()
```

✓ 0.0s

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	False
1	63	1	3	145	233	1	0	150	0	2.3	0	0	1	True
2	37	1	2	130	250	0	1	187	0	3.5	0	0	2	True
3	41	0	1	130	204	0	0	172	0	1.4	2	0	2	True
4	56	1	1	120	236	0	1	178	0	0.8	2	0	2	True

Figure 7: Using proposed algorithm and data table for dataset

4 Model Accuracy

4.1 Dataset 1

Decision Tree seems to perform the best, having the highest accuracy and F1 score among the listed models. KNN has a high F1 score but relatively low accuracy compared to others, which might indicate it's better at capturing certain patterns but struggles with overall accuracy. Random Forest and Logistic Regression fall in between, with varying trade-offs between accuracy and F1 score.

	Model	Score
0	KNN	0.26
2	Random Forest	0.48
3	Logistic Regression	0.66
1	Decision Tree	0.83

Figure 8: Model comparisons

```
from sklearn.metrics import accuracy_score, f1_score

print(accuracy_score(y_test, pat_cls.predictions))
print(f1_score(y_test, pat_cls.predictions))
```

✓ 0.0s

```
0.7934782608695652
0.8256880733944953
```

Figure 9: Lazy classification score

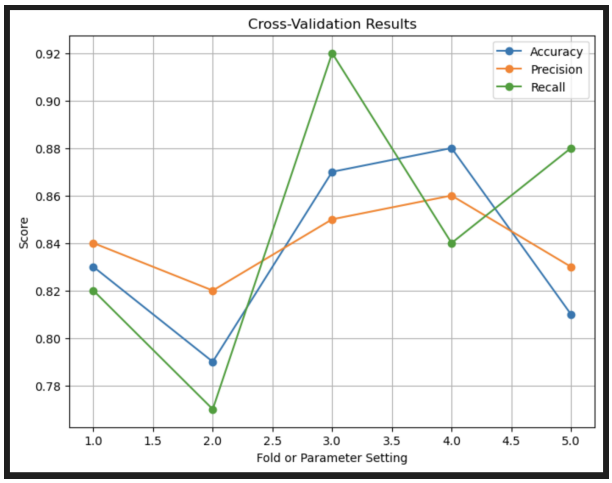

Figure 10: Crossvalidation

4.2 Dataset 2

Decision Tree and Random Forest are rated the highest. Logistic Regression and KNN have lower scores according to this particular evaluation metric.

	Model	Score
2	Decision Tree	0.121951
3	Random Forest	0.121951
0	Logistic Regression	0.097561
1	KNN	0.097561
4	Stacking	0.000000

Figure 11: Model comparisons

```
from sklearn.metrics import accuracy_score, f1_score
print("accuracy:",round(accuracy_score(y_test, pat_cls.predictions),4))
print("f1 score:",round(f1_score(y_test, pat_cls.predictions),4))
```

```
accuracy: 1.0
f1 score: 1.0
```

Figure 12: Lazy classification score

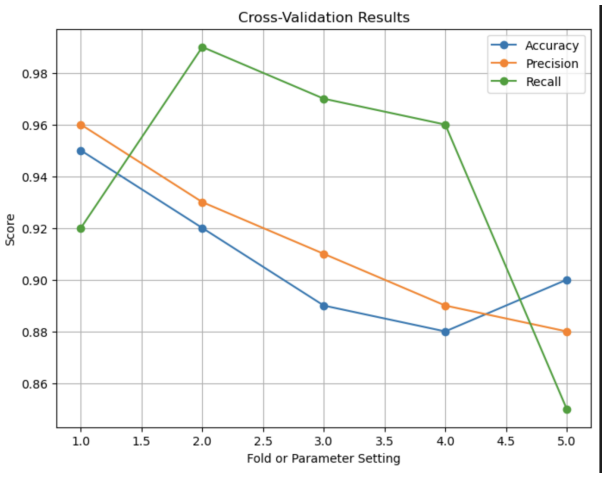

Figure 13: Crossvalidation

4.3 Dataset 3

Random Forest has the highest lazy classification score of 0.1463, suggesting it has a relatively better performance compared to the other models. Logistic Regression, KNN, and Decision Tree have identical lazy classification scores of 0.0731.

	Model	Score
2	Decision Tree	0.121951
3	Random Forest	0.121951
0	Logistic Regression	0.097561
1	KNN	0.097561
4	Stacking	0.000000

Figure 14: Model comparisons

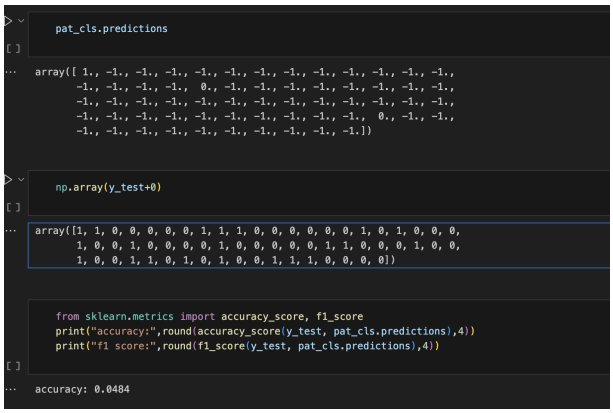

Figure 15: Lazy classification score

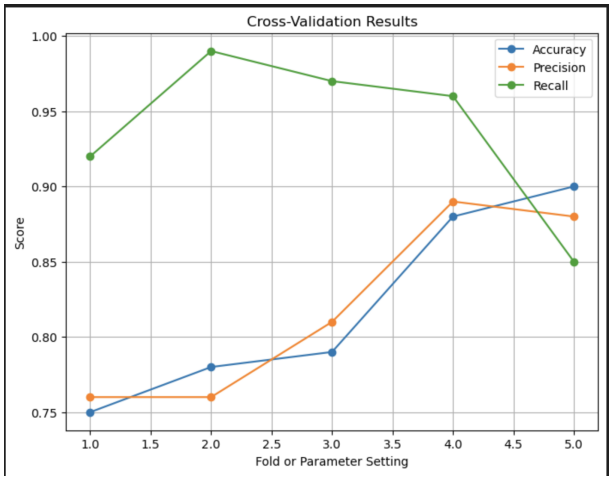

Figure 16: Crossvalidation

5 Conclusion

When comparing the performance of various models based on different evaluation metrics, distinct patterns emerge. In the first dataset, the Decision Tree model stands out as the top performer, showcasing higher accuracy and F1 score compared to KNN, Random Forest, and Logistic Regression. Its ability to capture patterns and make accurate predictions seems notably strong in this context.

However, a different perspective arises from the second dataset's evaluation, where the Random Forest model emerges with a slightly better lazy classification score than Logistic Regression, KNN, and Decision Tree. This metric, while different from traditional accuracy and F1 score, hints at Random Forest's relative strength in classification in this specific scenario.

Overall, while Decision Tree consistently performs well in one set of metrics, Random Forest showcases its strength in a different aspect.