December 1, 2022

ITM 885 Final Project

Classification Algorithms Comparison

Group 6: Talha Ali, Jorge Liakopulos, Di Ma

BROAD COLLEGE *of* BUSINESS

Table of Contents

Purpose of Project

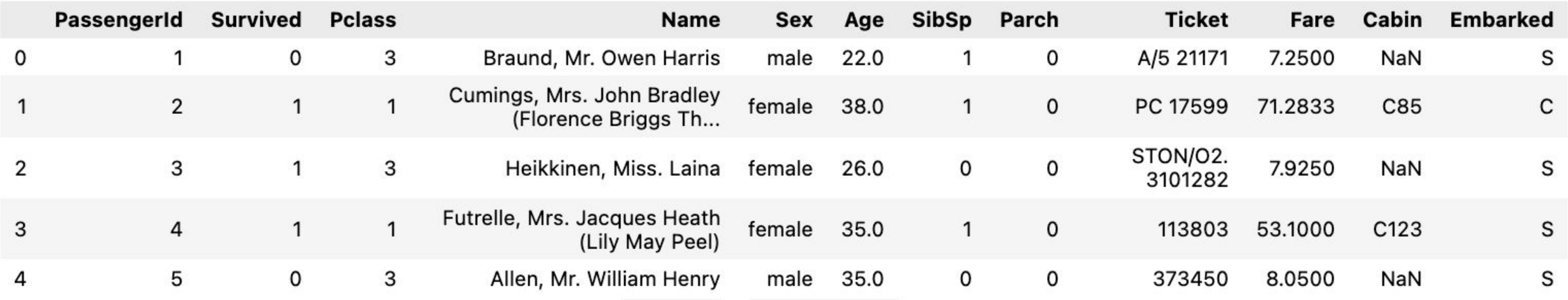
Classification Algorithms Comparison

Project Goal

Classification Algorithms Comparison

- Applying logistic regression, random forest, and SVM on the same dataset
- Comparing the performance
- Based on Titanic survival data

Dataset

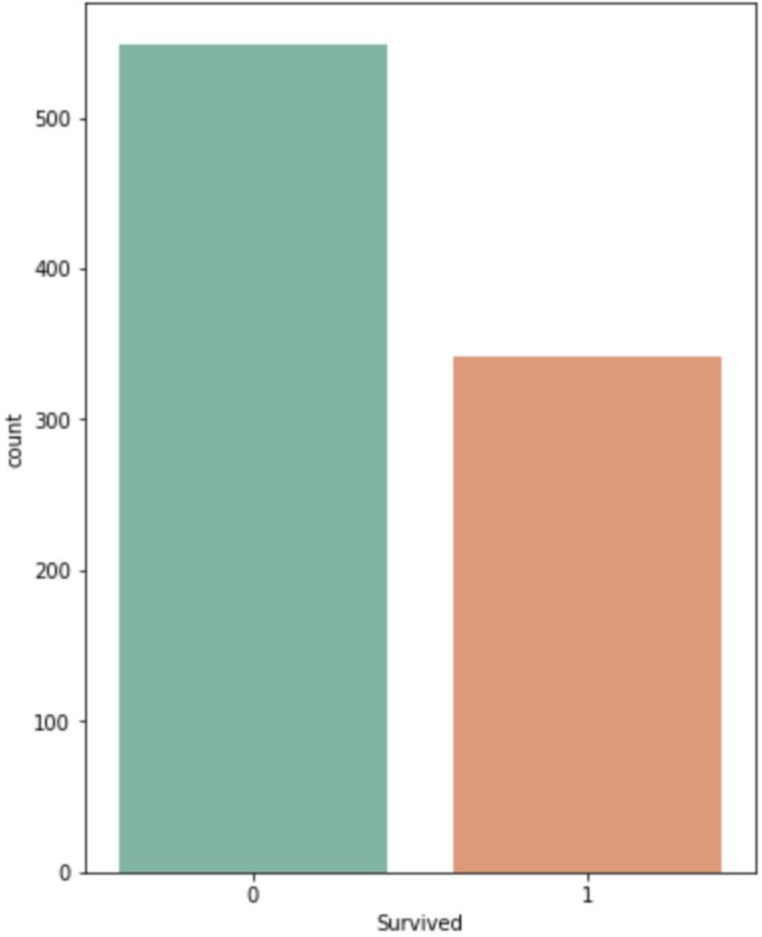


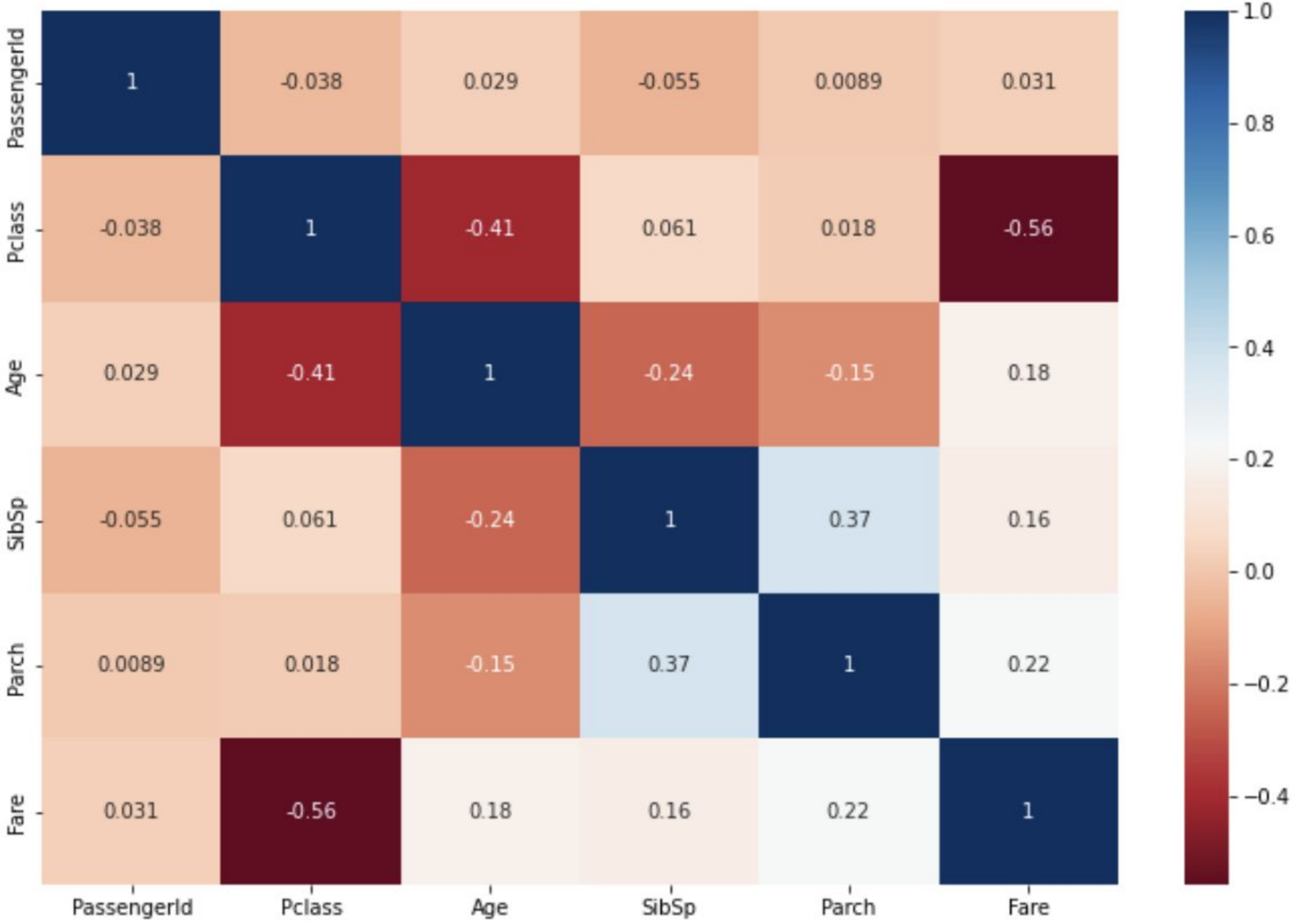
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Dataset

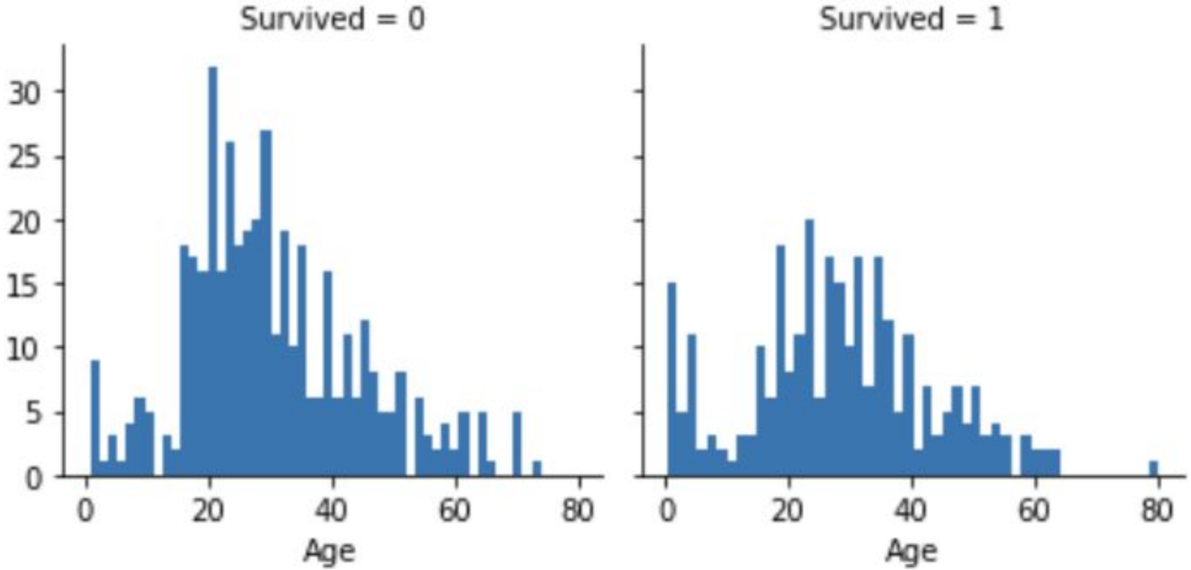
Passengers Survived the Titanic shipwreck

- What sorts of people were more likely to survive (0/1)
- 8 variables, including ticket class, sex, age, number of siblings/parents, fare, etc.
- Create models that predicts which passengers survived the Titanic shipwreck

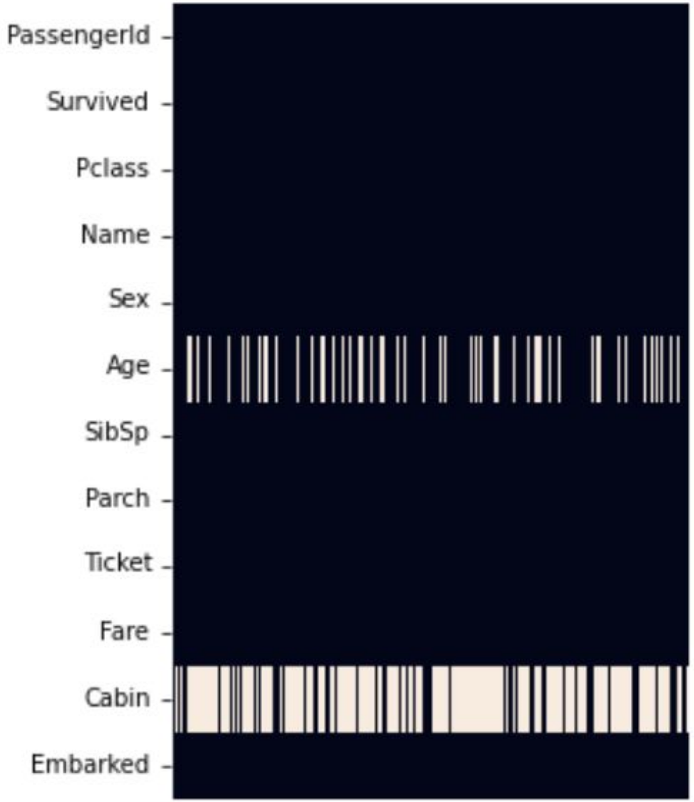

500
400
300
200
100
0
count
0
1
Survived



	PassengerId	Pclass	Age	SibSp	Parch	Fare
PassengerId	1	-0.038	0.029	-0.055	0.0089	0.031
Pclass	-0.038	1	-0.41	0.061	0.018	-0.56
Age	0.029	-0.41	1	-0.24	-0.15	0.18
SibSp	-0.055	0.061	-0.24	1	0.37	0.16
Parch	0.0089	0.018	-0.15	0.37	1	0.22
Fare	0.031	-0.56	0.18	0.16	0.22	1


Survived = 0
Survived = 1
0
5
10
15
20
25
30
0
20
40
60
80
Age
Age

White = Missing Values
Dark Blue = Non-Null



Feature Engineering

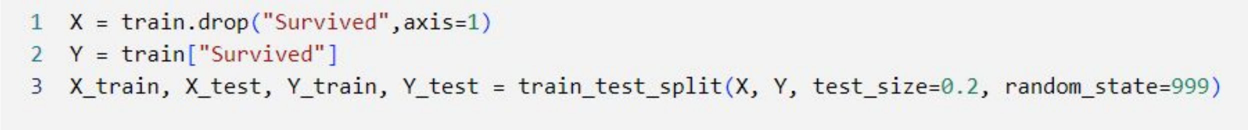
- Remove the column with more than 50% null values
- Fill the null values with mean
- Create new metrics by combining existing ones
    - For instance, create ‘Family size’ by combining parents and siblings variables

Random Forest

3.1 Random Forest

3.1.1 Dataset splitting

- Split the original dataset into train set and test set.

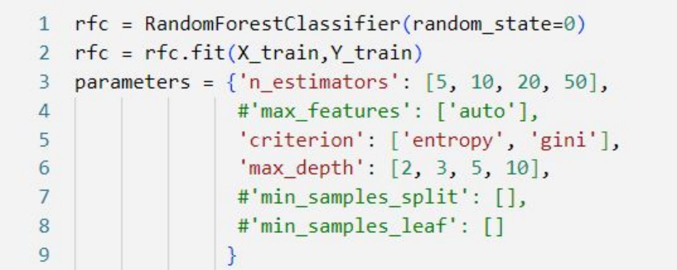


```
X = train.drop("Survived",axis=1)
Y = train["Survived"]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=999)
```

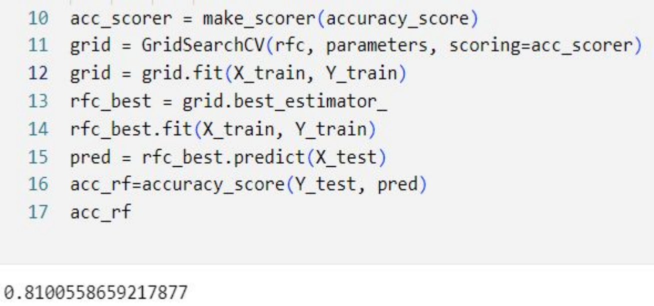
3.1 Random Forest

3.1.2 Tuning the hyper-parameters of an estimator

- Use grid search to find out the best parameters
    - Comment out some parameters to use default values.



```
rfc = RandomForestClassifier(random_state=0)
rfc = rfc.fit(X_train,Y_train)
parameters = {'n_estimators': [5, 10, 20, 50],
              #'max_features': ['auto'],
              'criterion': ['entropy', 'gini'],
              'max_depth': [2, 3, 5, 10],
              #'min_samples_split': [],
              #'min_samples_leaf': []
             }
```



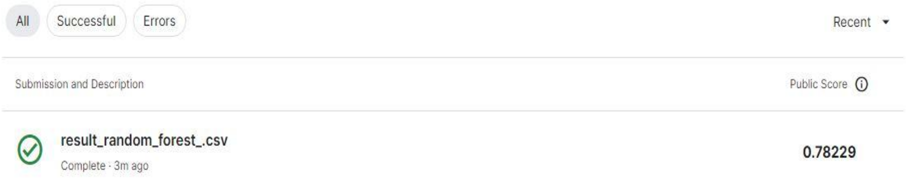
```
acc_scorer = make_scorer(accuracy_score)
grid = GridSearchCV(rfc, parameters, scoring=acc_scorer)
grid = grid.fit(X_train, Y_train)
rfc_best = grid.best_estimator_
rfc_best.fit(X_train, Y_train)
pred = rfc_best.predict(X_test)
acc_rf=accuracy_score(Y_test, pred)
acc_rf
```

```
0.8100558659217877
```

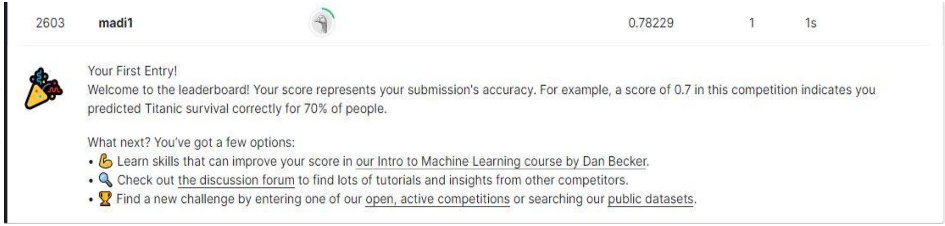
3.1 Random Forest

3.1.3 Performance

- 78% Accuracy of final result
- Top 18% ranking among all 13,000+ teams



Submission and Description	Public Score
result_random_forest_.csv Complete · 3m ago	0.78229



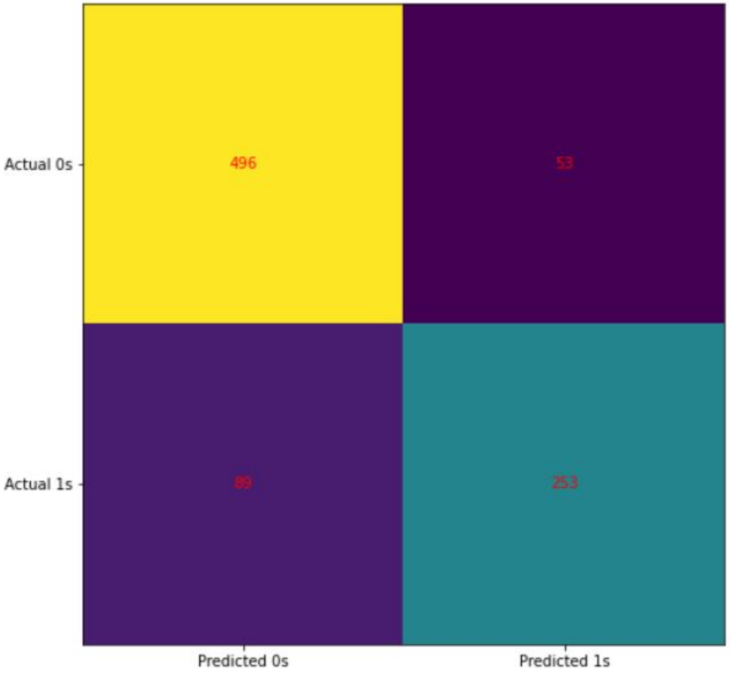
```
2603    madi1    0.78229    1    1s
```

Your First Entry!
Welcome to the leaderboard! Your score represents your submission's accuracy. For example, a score of 0.7 in this competition indicates you predicted Titanic survival correctly for 70% of people.

What next? You've got a few options:
- 💪 Learn skills that can improve your score in our Intro to Machine Learning course by Dan Becker.
- 🔍 Check out the discussion forum to find lots of tutorials and insights from other competitors.
- 🏆 Find a new challenge by entering one of our open, active competitions or searching our public datasets.

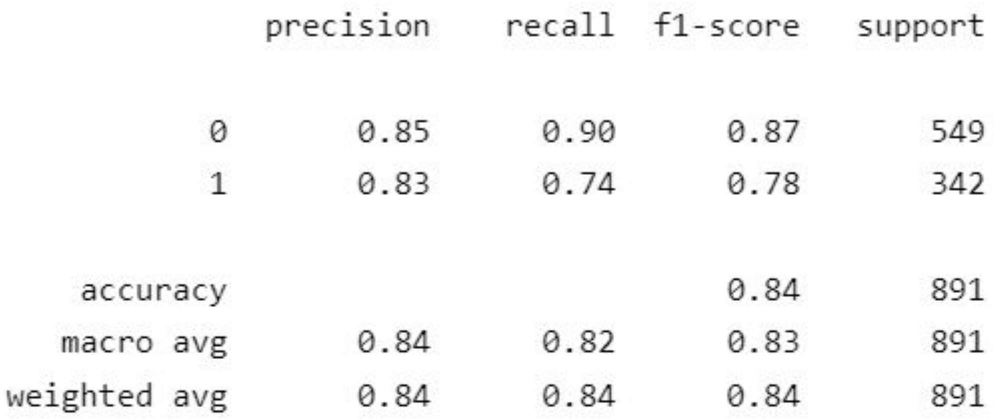
3.1 Random Forest

3.1.3 Performance



3.1 Random Forest

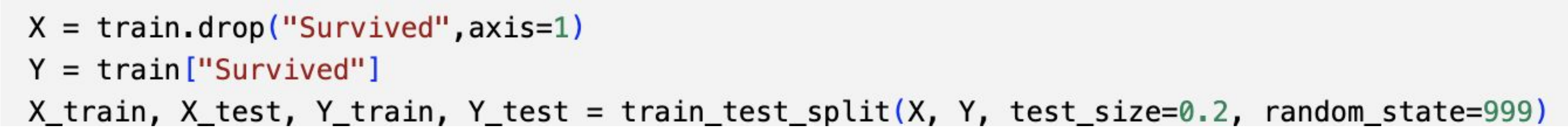
3.1.3 Performance



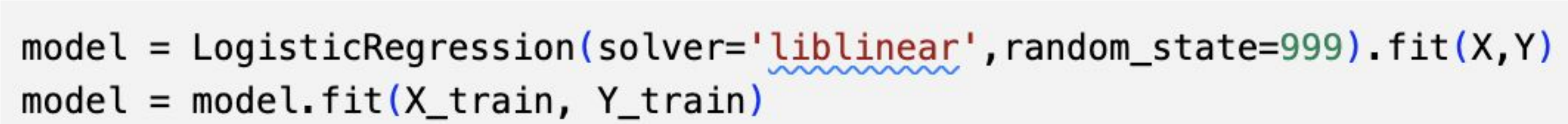
```
              precision    recall  f1-score   support

           0       0.85      0.90      0.87       549
           1       0.83      0.74      0.78       342

    accuracy                           0.84       891
   macro avg       0.84      0.82      0.83       891
weighted avg       0.84      0.84      0.84       891
```

Logistic Regression

3.2 Logistic Regression

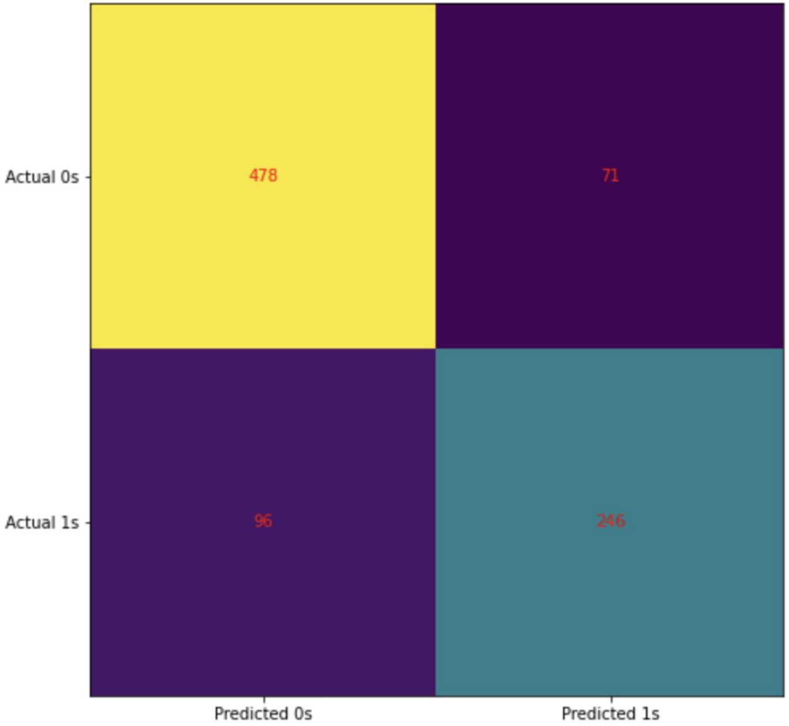


```
X = train.drop("Survived",axis=1)
Y = train["Survived"]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=999)
```



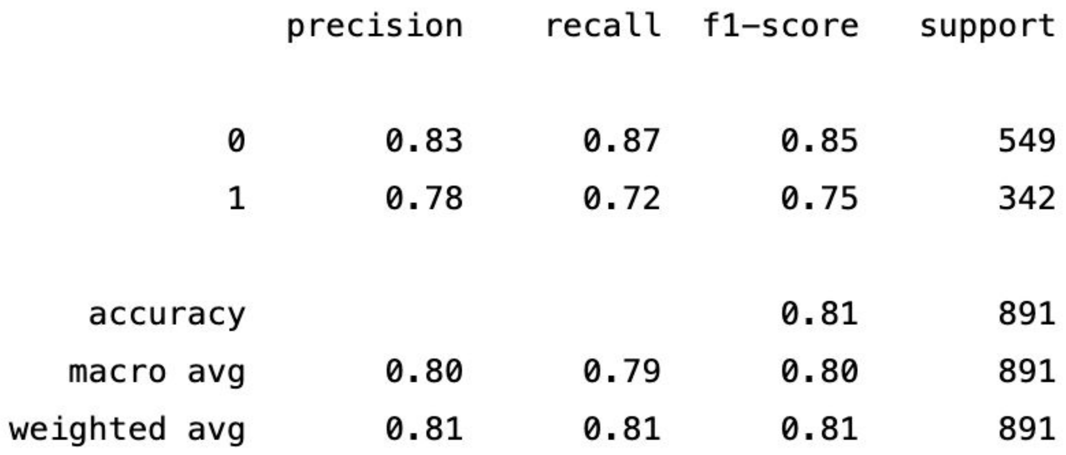
```
model = LogisticRegression(solver='liblinear',random_state=999).fit(X,Y)
model = model.fit(X_train, Y_train)
```

3.3 Logistic Regression



	Predicted 0s	Predicted 1s
Actual 0s	478	71
Actual 1s	96	246

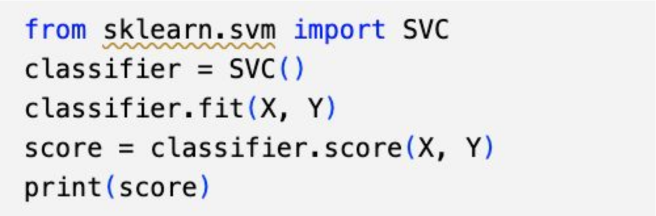
3.3 Logistic Regression



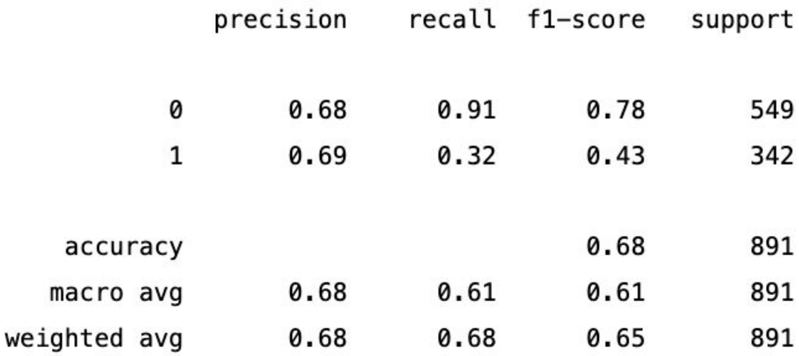
```
              precision    recall  f1-score   support

           0       0.83      0.87      0.85       549
           1       0.78      0.72      0.75       342

    accuracy                           0.81       891
   macro avg       0.80      0.79      0.80       891
weighted avg       0.81      0.81      0.81       891
```

SVM

3.4 SVM



```python
from sklearn.svm import SVC
classifier = SVC()
classifier.fit(X, Y)
score = classifier.score(X, Y)
print(score)
```



```
              precision    recall  f1-score   support

           0       0.68      0.91      0.78       549
           1       0.69      0.32      0.43       342

    accuracy                           0.68       891
   macro avg       0.68      0.61      0.61       891
weighted avg       0.68      0.68      0.65       891
```