DSAI Individual Project

First of all, for this project, we will use the library pandas imported as pd.

1. ***Classify images of flowers based on their features such as the petal length, petal width, sepal length, and sepal width? Use the Iris Flowers dataset, which contains information on 150 iris flowers belonging to three different species, and apply SVM or random forest to classify the flowers.***

To start, we have to write in python the commands that will give us the elements to classify all these flowers. First, we have to open the Database in python using this command :

```
db = pd.read_csv('IRIS_ Flower_Dataset.csv')
```

Then , we have to classify each species of flower by each criterion. for this, we will use these commands :

```
db_setosa = db.loc[db['species'] == 'Iris-setosa']
```
*

with this command, I select in the database only the flowers that are from "Iris-setosa" specie.

```
print(db_setosa.sort_values(by='sepal_length', ascending=False).head(10))
```

with this command, we print the 10 top values of the criteria chosen before. here, the criteria is 'sepal_length', we will do this again with sepal width, petal length and petal width. When we do it with all the species, we get theses tables :

	sepal_length	sepal_width	petal_length	petal_width	species
14	5.8	4.0	1.2	0.2	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
36	5.5	3.5	1.3	0.2	Iris-setosa
33	5.5	4.2	1.4	0.2	Iris-setosa
31	5.4	3.4	1.5	0.4	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
20	5.4	3.4	1.7	0.2	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa

----

	sepal_length	sepal_width	petal_length	petal_width	species
15	5.7	4.4	1.5	0.4	Iris-setosa
33	5.5	4.2	1.4	0.2	Iris-setosa
32	5.2	4.1	1.5	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
46	5.1	3.8	1.6	0.2	Iris-setosa
44	5.1	3.8	1.9	0.4	Iris-setosa

	sepal_length	sepal_width	petal_length	petal_width	species
24	4.8	3.4	1.9	0.2	Iris-setosa
44	5.1	3.8	1.9	0.4	Iris-setosa
23	5.1	3.3	1.7	0.5	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
20	5.4	3.4	1.7	0.2	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
25	5.0	3.0	1.6	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
26	5.0	3.4	1.6	0.4	Iris-setosa
30	4.8	3.1	1.6	0.2	Iris-setosa

----

	sepal_length	sepal_width	petal_length	petal_width	species
43	5.0	3.5	1.6	0.6	Iris-setosa
23	5.1	3.3	1.7	0.5	Iris-setosa
26	5.0	3.4	1.6	0.4	Iris-setosa
31	5.4	3.4	1.5	0.4	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
44	5.1	3.8	1.9	0.4	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
21	5.1	3.7	1.5	0.4	Iris-setosa
40	5.0	3.5	1.3	0.3	Iris-setosa

```
    sepal_length  sepal_width  petal_length  petal_width          species
50           7.0          3.2           4.7          1.4  Iris-versicolor
52           6.9          3.1           4.9          1.5  Iris-versicolor
76           6.8          2.8           4.8          1.4  Iris-versicolor
86           6.7          3.1           4.7          1.5  Iris-versicolor
77           6.7          3.0           5.0          1.7  Iris-versicolor
65           6.7          3.1           4.4          1.4  Iris-versicolor
58           6.6          2.9           4.6          1.3  Iris-versicolor
75           6.6          3.0           4.4          1.4  Iris-versicolor
54           6.5          2.8           4.6          1.5  Iris-versicolor
74           6.4          2.9           4.3          1.3  Iris-versicolor

--------------------------------------------------

    sepal_length  sepal_width  petal_length  petal_width          species
85           6.0          3.4           4.5          1.6  Iris-versicolor
56           6.3          3.3           4.7          1.6  Iris-versicolor
50           7.0          3.2           4.7          1.4  Iris-versicolor
51           6.4          3.2           4.5          1.5  Iris-versicolor
70           5.9          3.2           4.8          1.8  Iris-versicolor
65           6.7          3.1           4.4          1.4  Iris-versicolor
52           6.9          3.1           4.9          1.5  Iris-versicolor
86           6.7          3.1           4.7          1.5  Iris-versicolor
95           5.7          3.0           4.2          1.2  Iris-versicolor
91           6.1          3.0           4.6          1.4  Iris-versicolor

--------------------------------------------------


    sepal_length  sepal_width  petal_length  petal_width          species
83           6.0          2.7           5.1          1.6  Iris-versicolor
77           6.7          3.0           5.0          1.7  Iris-versicolor
72           6.3          2.5           4.9          1.5  Iris-versicolor
52           6.9          3.1           4.9          1.5  Iris-versicolor
70           5.9          3.2           4.8          1.8  Iris-versicolor
76           6.8          2.8           4.8          1.4  Iris-versicolor
86           6.7          3.1           4.7          1.5  Iris-versicolor
63           6.1          2.9           4.7          1.4  Iris-versicolor
73           6.1          2.8           4.7          1.2  Iris-versicolor
50           7.0          3.2           4.7          1.4  Iris-versicolor

--------------------------------------------------

    sepal_length  sepal_width  petal_length  petal_width          species
70           5.9          3.2           4.8          1.8  Iris-versicolor
77           6.7          3.0           5.0          1.7  Iris-versicolor
56           6.3          3.3           4.7          1.6  Iris-versicolor
85           6.0          3.4           4.5          1.6  Iris-versicolor
83           6.0          2.7           5.1          1.6  Iris-versicolor
86           6.7          3.1           4.7          1.5  Iris-versicolor
54           6.5          2.8           4.6          1.5  Iris-versicolor
84           5.4          3.0           4.5          1.5  Iris-versicolor
68           6.2          2.2           4.5          1.5  Iris-versicolor
78           6.0          2.9           4.5          1.5  Iris-versicolor
```

	sepal_length	sepal_width	petal_length	petal_width	species
131	7.9	3.8	6.4	2.0	Iris-virginica
135	7.7	3.0	6.1	2.3	Iris-virginica
118	7.7	2.6	6.9	2.3	Iris-virginica
122	7.7	2.8	6.7	2.0	Iris-virginica
117	7.7	3.8	6.7	2.2	Iris-virginica
105	7.6	3.0	6.6	2.1	Iris-virginica
130	7.4	2.8	6.1	1.9	Iris-virginica
107	7.3	2.9	6.3	1.8	Iris-virginica
125	7.2	3.2	6.0	1.8	Iris-virginica
129	7.2	3.0	5.8	1.6	Iris-virginica

----

	sepal_length	sepal_width	petal_length	petal_width	species
131	7.9	3.8	6.4	2.0	Iris-virginica
117	7.7	3.8	6.7	2.2	Iris-virginica
109	7.2	3.6	6.1	2.5	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
136	6.3	3.4	5.6	2.4	Iris-virginica
100	6.3	3.3	6.0	2.5	Iris-virginica
144	6.7	3.3	5.7	2.5	Iris-virginica
124	6.7	3.3	5.7	2.1	Iris-virginica
115	6.4	3.2	5.3	2.3	Iris-virginica
143	6.8	3.2	5.9	2.3	Iris-virginica

	sepal_length	sepal_width	petal_length	petal_width	species
118	7.7	2.6	6.9	2.3	Iris-virginica
117	7.7	3.8	6.7	2.2	Iris-virginica
122	7.7	2.8	6.7	2.0	Iris-virginica
105	7.6	3.0	6.6	2.1	Iris-virginica
131	7.9	3.8	6.4	2.0	Iris-virginica
107	7.3	2.9	6.3	1.8	Iris-virginica
135	7.7	3.0	6.1	2.3	Iris-virginica
130	7.4	2.8	6.1	1.9	Iris-virginica
109	7.2	3.6	6.1	2.5	Iris-virginica
100	6.3	3.3	6.0	2.5	Iris-virginica

----

	sepal_length	sepal_width	petal_length	petal_width	species
100	6.3	3.3	6.0	2.5	Iris-virginica
144	6.7	3.3	5.7	2.5	Iris-virginica
109	7.2	3.6	6.1	2.5	Iris-virginica
114	5.8	2.8	5.1	2.4	Iris-virginica
140	6.7	3.1	5.6	2.4	Iris-virginica
136	6.3	3.4	5.6	2.4	Iris-virginica
141	6.9	3.1	5.1	2.3	Iris-virginica
120	6.9	3.2	5.7	2.3	Iris-virginica
143	6.8	3.2	5.9	2.3	Iris-virginica
115	6.4	3.2	5.3	2.3	Iris-virginica

After the analyses of these tables, we can conclude that :

- the Iris-virginia is the specie that have the biggest sepal length mean, followed by Iris-versicolor and then the Iris-setosa

- the Iris-setosa is the specie that have the biggest sepal width mean, followed by Iris-virginica and then the Iris-versicolor

- the Iris-virginia is the specie that have the biggest petal length mean, followed by Iris-versicolor and then the Iris-setosa

- the Iris-virginia is the specie that have the biggest petal width mean, followed by Iris-setosa and then the Iris-versicolor

Now, we have to verify the accuracy of our model. To do this, we will use the library sklearn, and import these methods :

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
```

First of all, we will target the column that interests us. In this case, it's the column "species". So we write :

```
X_train, X_test, y_train, y_test =
train_test_split(db.drop(['species'],axis='columns'),db['species'],test_size=0.2)
```

then, we need to choose a model. We will here use the RandomForest Model because of its precision, simplicity and flexibility.

```
model = RandomForestClassifier()
```

We will train our model using this method :

```
model.fit(X_train, y_train)
```

trained, our model give us this score :

```
model.score(X_test,y_test)
```

```
0.9666666666666667
```

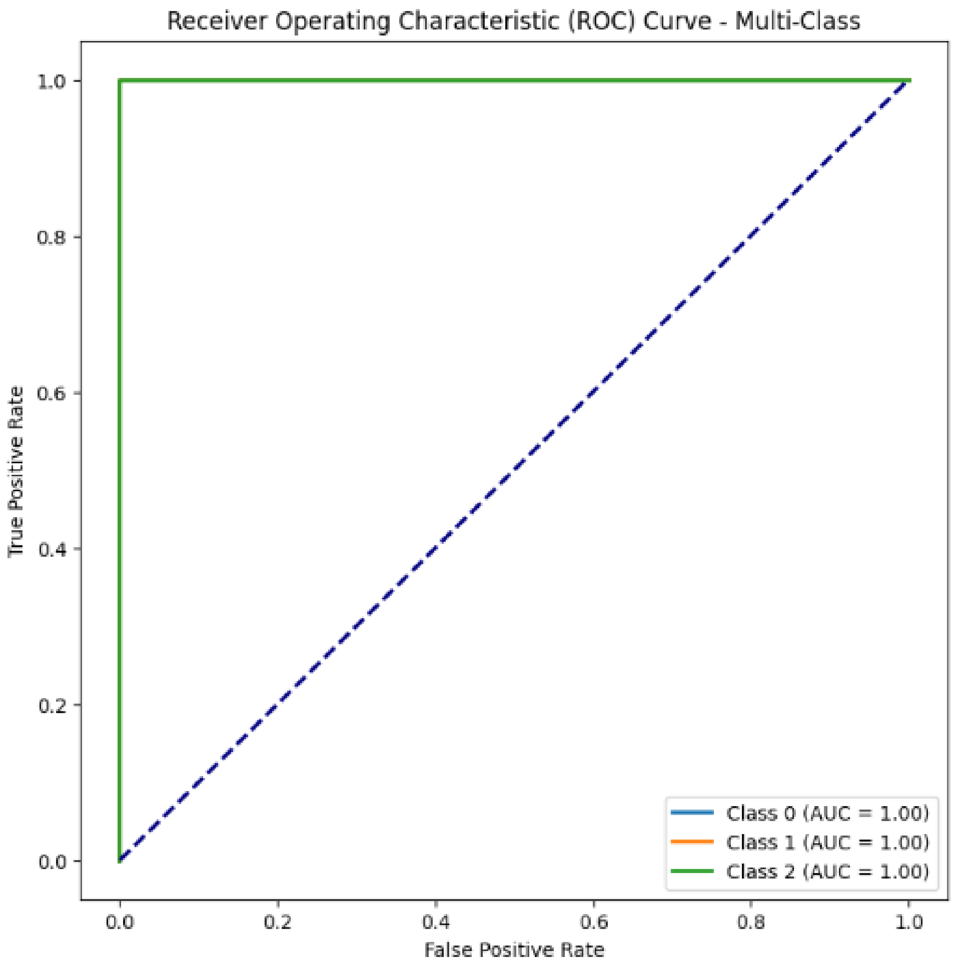
To finish, we have to predict the Y value from the X Value, and do a classification report, using sklearn.metrics :

```
from sklearn.metrics import  accuracy_score
y_pred_test = model.predict(X_test)
print(classification_report(y_test, y_pred_test))
```

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00         6
Iris-versicolor       0.89      1.00      0.94         8
 Iris-virginica       1.00      0.94      0.97        16

       accuracy                           0.97        30
      macro avg       0.96      0.98      0.97        30
   weighted avg       0.97      0.97      0.97        30
```

We can see in this report the precision, recall, f1-score and support for each specie of flower.

finally, we have to print the ROC and AUC graph :



2. ***Predict the survival of passengers on the Titanic based on their age, sex, class, and other features? Use the Titanic dataset, which contains information about passengers on the Titanic, and apply logistic regression to predict survival.***

To start, we want to know how many passengers survived, and we can do it using this command :

```
db = db_train.loc[db_train['Survived'] == 1]
```

when we print it, we get :

```
     PassengerId  Survived  Pclass  \
1              2         1       1
573          574         1       3
591          592         1       1
587          588         1       1
585          586         1       1
..           ...       ...     ...
306          307         1       1
305          306         1       1
303          304         1       2
301          302         1       3
889          890         1       1

                                                  Name     Sex    Age  SibSp  \
1    Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.00      1
573                                  Kelly, Miss. Mary  female    NaN      0
591    Stephenson, Mrs. Walter Bertram (Martha Eustis)  female  52.00      1
587                   Frolicher-Stehli, Mr. Maxmillian    male  60.00      1
585                                Taussig, Miss. Ruth  female  18.00      0
..                                                 ...     ...    ...    ...
306                            Fleming, Miss. Margaret  female    NaN      0
305                     Allison, Master. Hudson Trevor    male   0.92      1
303                                Keane, Miss. Nora A  female    NaN      0
301                                 McCoy, Mr. Bernard    male    NaN      2
889                              Behr, Mr. Karl Howell    male  26.00      0

     Parch    Ticket      Fare    Cabin Embarked
1        0  PC 17599   71.2833      C85        C
573      0     14312    7.7500      NaN        Q
591      0     36947   78.2667      D20        C
587      1     13567   79.2000      B41        C
585      2    110413   79.6500      E68        S
..     ...       ...       ...      ...      ...
306      0     17421  110.8833      NaN        C
305      2    113781  151.5500  C22 C26        S
303      0    226593   12.3500     E101        Q
301      0    367226   23.2500      NaN        Q
889      0    111369   30.0000     C148        C

[342 rows x 12 columns]
```

we see that there is 342 rows, and so 341 passengers that survived on 891 in total.

now, we want to see if there is a majority of men or women. For this, we use this method :

```
db_m = db.loc[db['Sex'] == 'male']
db_f = db.loc[db['Sex'] == 'female']
```

we get this :

```
     PassengerId  Survived  Pclass                              Name   Sex  \
17            18         1       2      Williams, Mr. Charles Eugene  male
21            22         1       2              Beesley, Mr. Lawrence  male
23            24         1       1      Sloper, Mr. William Thompson  male
36            37         1       3                 Mamee, Mr. Hanna  male
55            56         1       1                 Woolner, Mr. Hugh  male
..           ...       ...     ...                               ...   ...
838          839         1       3                  Chip, Mr. Chang  male
839          840         1       1             Marechal, Mr. Pierre  male
857          858         1       1              Daly, Mr. Peter Denis  male
869          870         1       3  Johnson, Master. Harold Theodor  male
889          890         1       1             Behr, Mr. Karl Howell  male

      Age  SibSp  Parch  Ticket     Fare Cabin Embarked
17    NaN      0      0  244373  13.0000   NaN        S
21   34.0      0      0  248698  13.0000   D56        S
23   28.0      0      0  113788  35.5000    A6        S
36    NaN      0      0    2677   7.2292   NaN        C
55    NaN      0      0   19947  35.5000   C52        S
..    ...    ...    ...     ...      ...   ...      ...
838  32.0      0      0    1601  56.4958   NaN        S
839   NaN      0      0   11774  29.7000   C47        C
857  51.0      0      0  113055  26.5500   E17        S
869   4.0      1      1  347742  11.1333   NaN        S
889  26.0      0      0  111369  30.0000  C148        C

[109 rows x 12 columns]
```

```
                                                   Name     Sex   Age  SibSp  \
1     Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                                Heikkinen, Miss. Laina  female  26.0      0
3          Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
8     Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0
9                   Nasser, Mrs. Nicholas (Adele Achem)  female  14.0      1
..                                                  ...     ...   ...    ...
874               Abelson, Mrs. Samuel (Hannah Wizosky)  female  28.0      1
875                   Najib, Miss. Adele Kiamie "Jane"  female  15.0      0
879        Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)  female  56.0      0
880         Shelley, Mrs. William (Imanita Parrish Hall)  female  25.0      0
887                        Graham, Miss. Margaret Edith  female  19.0      0

     Parch            Ticket     Fare Cabin Embarked
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
8        2            347742  11.1333   NaN        S
9        0            237736  30.0708   NaN        C
..     ...               ...      ...   ...      ...
874      0         P/PP 3381  24.0000   NaN        C
875      0              2667   7.2250   NaN        C
879      1             11767  83.1583   C50        C
880      1            230433  26.0000   NaN        S
887      0            112053  30.0000   B42        S

[233 rows x 12 columns]
```

We see here that 232 women have survived, while for the men there are only 108 survivors. So we can say that women are more likely to survive than men.

Now, we have to see if the class is a criterion for the survival

for this, we use :

```
db_1 = db.loc[db['Pclass'] == 1]
db_2 = db.loc[db['Pclass'] == 2]
db_3 = db.loc[db['Pclass'] == 3]
```

By printing this, we obtain 135 survivors from class 1, 86 from class 2 and 118 from class 3

In total, we have 215 passengers in class 1, 283 passengers in class 2 and 490 passengers in class 3

we get 63% of survivors in class 1, 30% in class 2 and 24% for class 3. That seems logical because first class is the expensive one, and consequently it have to be the safest for the client.

We will now see if age influences survival.
We have a lot of missing values, so we will replace them with the mean of the values that we get using this method :

```
mean_age = db['Age'].mean()
```

```
db_train['Age'].fillna(mean_age, inplace=True)
db['Age'].fillna(mean_age, inplace=True)
```

we get a mean age of 28.34 years.
we get also a median of 28, that means that we have 50% of the passengers that have over than 28 years, and 50% below.

so, we have 529 persons that have 28 years or more, and 529 that have 28 years or less (we count 2 times those that have 28 years)

for the persons that have 28 years or less, we have 193 survivors, while for those that have 28 years or more, we have 199 survivors. We can conclude that age doesn't matter here.

Now, to be able to use the logistic regression, we have to drop from our database the columns that are uninteresting using the method drop :

```
db_train.drop(columns = 'Embarked',inplace=True)
```

then, we will replace the column " Sex" by a column where the males will be marked as a 1 and the females by a 0, binary.

```
db_train['Sex'] = db_train['Sex'].map({'male': 1, 'female': 0})
```

we get this :

```
     PassengerId  Survived  Pclass  Sex   Age  SibSp  Parch     Fare
0              1         0       3    1  22.0      1      0   7.2500
1              2         1       1    0  38.0      1      0  71.2833
2              3         1       3    0  26.0      0      0   7.9250
3              4         1       1    0  35.0      1      0  53.1000
4              5         0       3    1  35.0      0      0   8.0500
..           ...       ...     ...  ...   ...    ...    ...      ...
886          887         0       2    1  27.0      0      0  13.0000
887          888         1       1    0  19.0      0      0  30.0000
888          889         0       3    0   NaN      1      2  23.4500
889          890         1       1    1  26.0      0      0  30.0000
890          891         0       3    1  32.0      0      0   7.7500

[891 rows x 8 columns]
```

Then, finally, I will replace the columns “SibSp” and “Parch” with a column “Family”, To simplify the table, using this :

```
db_train['Family'] = db_train.apply(lambda row: 1 if row['SibSp'] == 1 or row['Parch'] == 1 else 0, axis=1)
db_g_test['Family'] = db_g_test.apply(lambda row: 1 if row['SibSp'] == 1 or row['Parch'] == 1 else 0, axis=1)
```

from this, we will use the logistic regression model.

my code for it is this :

```
X_train = db_train.drop(columns=['Survived','SibSp','Parch'])
y_train = db_train['Survived']

X_test = db_g_test.drop(columns=['PassengerId','Survived','SibSp','Parch'])
model = LogisticRegression()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)


accuracy = accuracy_score(db_g_test['Survived'], y_pred)
classification_report_result = classification_report(db_g_test['Survived'], y_pred)

print(f'Accuracy: {accuracy}')
print('Classification Report:')
print(classification_report_result)
```

db_g_test is a merged database between the gender_submission database and the test database.

With this code, I obtain this :

```
Accuracy: 0.9521531100478469
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.96      0.96       266
           1       0.93      0.94      0.93       152

    accuracy                           0.95       418
   macro avg       0.95      0.95      0.95       418
weighted avg       0.95      0.95      0.95       418
```

Our model has an accuracy of 95,21%. Let's now see if optimising hyperparameters can help us to upgrade our accuracy. For this, we have to import a new method :

```
from sklearn.model_selection import GridSearchCV
```

with this, we can write this code :

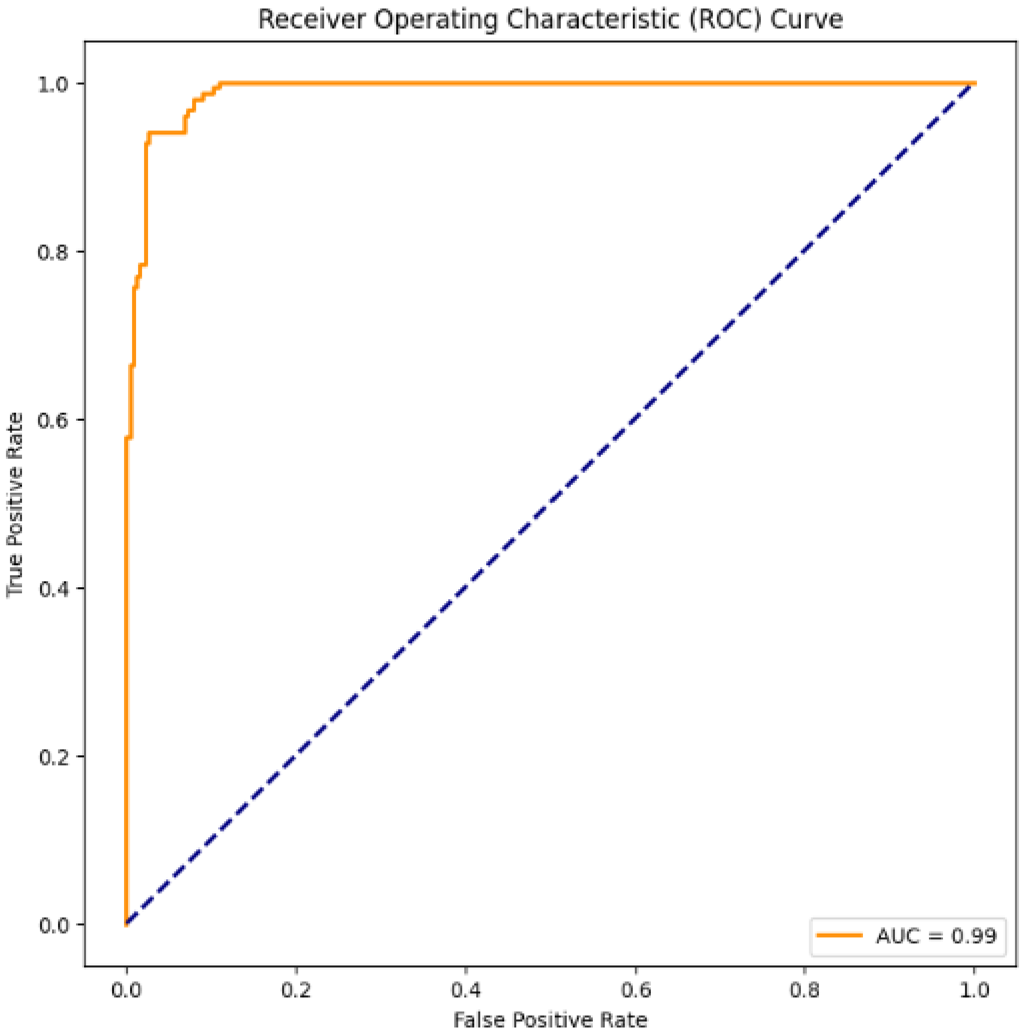
```
param_grid = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'solver': ['liblinear', 'newton-cg', 'lbfgs', 'sag', 'saga'] ,
}

grid_search = GridSearchCV(model, param_grid, cv=5, scoring='accuracy', verbose=1)
grid_search.fit(X_train, y_train)


best_params = grid_search.best_params_
print("Bests Hyperparameters : ", best_params)

best_model = grid_search.best_estimator_
best_model.fit(X_train,y_train)
y_best_pred = model.predict(X_test)


accuracy = accuracy_score(db_g_test['Survived'], y_best_pred)
print(accuracy)
```

with this, we obtain :

```
Bests Hyperparameters :  {'C': 1, 'solver': 'liblinear'}
0.9521531100478469
```

It means that, with these hyperparameters, I get the most precise model possible..

Now, to conclude, we will print the ROC and AUC graphs :



We have an AUC of 99%, very close to the perfect classifier. That means that our model is reliable and that we can trust it.