Supervised Machine Learning : Classification Final Project

The dataset to be used in this project is the Titanic dataset.

Purpose

A major accident that hit an iceberg in 1912 killed 2224 passengers and 1502 crew members. This accident resulted in increased safety measures on ships around the world. Because there were not enough lifeboats on board in this accident. Looking at the hierarchical classes on the ship, there is a relationship between people's financial means, gender and age according to their survival status. **In this project, our aim is to classify the passengers on board the titanic ship as Died or Survived according to their specific characteristics.**

Features of this dataset

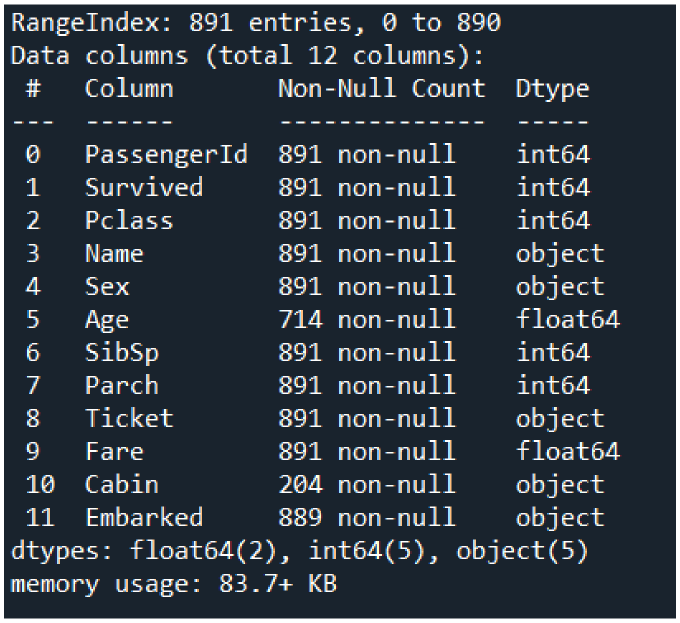
- Survived : Shows the life and death status of the passenger. (0 : Died , 1 : Survived)
- PassengerID : Passenger number
- Name : Passenger name
- Pclass : Passenger Class (1 : First Class , 2: Second Class , 3 : Third Class)

- Sex : Gender (Male or Female)
- SibSp : Number of siblings and spouses of the passenger on board
- Parch : Number of parents or children on board of the passenger.
- Ticket : Ticket number
- Fare : Passenger’s payment
- Cabin : Cabin number
- Embarked : Embarkation port (C : Cherbourg, Q : Queenstown, S : Southampton)

Examining the dataset

The data.info() command can be used to examine the dataset.

When we write this command in Python, we get the following table as output



```
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

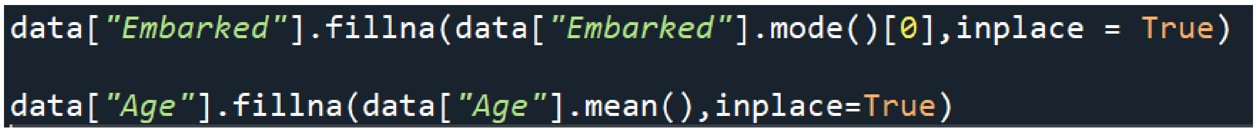
When we examined the dataset, we saw that some of the age , cabin and embarked data were missing. To avoid this issue, we can fill in missing data in our dataset or remove columns with missing data.

Filling in missing data will be more useful in terms of not losing data.

The mean age in the data set can be assigned to the blank data to fill in the gaps in the age data.

The gaps in column a can be replaced with the most embarked port.

The python code to be used for this is as follows.

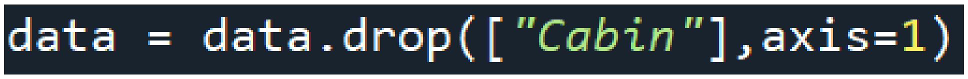


```
data["Embarked"].fillna(data["Embarked"].mode()[0],inplace = True)

data["Age"].fillna(data["Age"].mean(),inplace=True)
```

Cabin column can be removed from the data set because it contains too many variables and too many empty data.

The python code to be used for this is as follows.



```
data = data.drop(["Cabin"],axis=1)
```

Since there is no empty data in the data set, feature selection can be started.

Choosing the features to use

The column to be estimated in this series is the survived column.

Since the Passenger ID column is independent of the dataset, it will be omitted to prevent overfitting.

The ticket column can be omitted from the dataset as it contains many different variables.

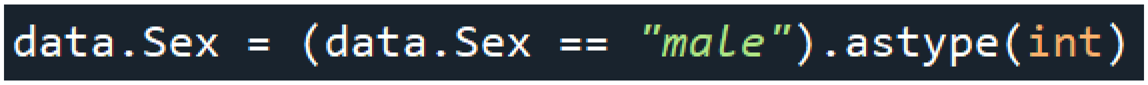
Since each of the variables in the name column are different from each other, the name column can be omitted from the data set.

Converting categorical data to numerical data

Since machine learning algorithms make predictions on numerical data types, categorical data types must be represented as numerical data. We can use some commands in python for this.

The categorical data in the data set are sex and embarked.

The following python code can be used to display gender data as 0 and 1.

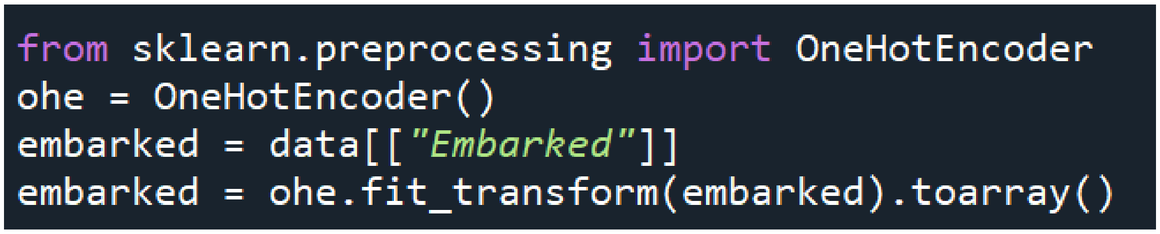


```
data.Sex = (data.Sex == "male").astype(int)
```

The astype(int) command will show the genders as 0 and 1, not True or false.

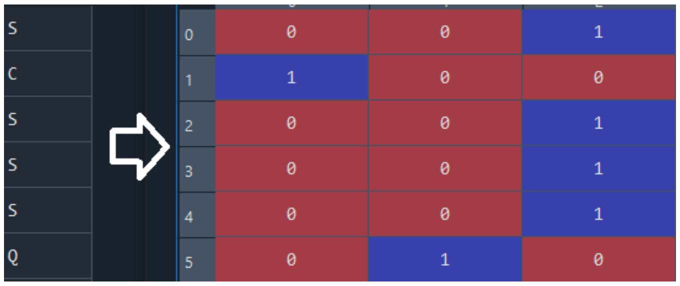
OneHotEncoder can be used to make the embarked column numerical.

The python code to be used for this is as follows



```python
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder()
embarked = data[["Embarked"]]
embarked = ohe.fit_transform(embarked).toarray()
```

OneHotEncoder converts data to numeric data type as in the figure



S
C
S
S
S
Q

0	0	0	1
1	1	0	0
2	0	0	1
3	0	0	1
4	0	0	1
5	0	1	0

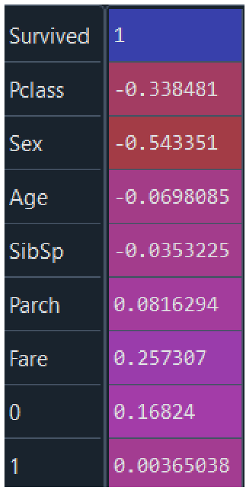
Dummy veriable trap

A dummy variable trap can be an issue that negatively impacts correlation. To avoid this, one of the columns with the same meaning can be omitted. For example, if a person is not a man, it is a woman. Therefore, using both male and female columns at the same time is unnecessary and risky for the algorithm. Using n-1 columns out of n columns is sufficient to prevent this.

In other words, it will be sufficient to use 2 columns from the embarked column converted to numeric.

Correlation matrix

When we look at the correlation between the features and the Survived column, we got the following table as output.



Survived	1
Pclass	-0.338481
Sex	-0.543351
Age	-0.0698085
SibSp	-0.0353225
Parch	0.0816294
Fare	0.257307
0	0.16824
1	0.00365038

r being the correlation coefficient

1. If r<|0.2l, there is very weak correlation or no correlation.
2. Weak correlation between l0.2-0.4l.
3. Moderate correlation between l0.4-0.6l.
4. High correlation between l0.6-0.8l.
5. If l0.8l>r, it is interpreted as a very high correlation.

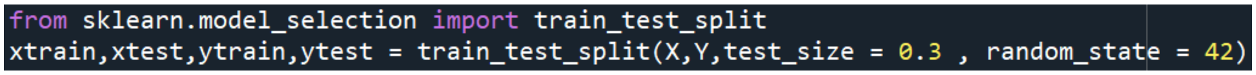
After selecting the appropriate data according to the correlation matrix, the data can be separated.

Separating the data set into training and testing

Let's define the featured dataset as X, and the dataset with the prediction data, namely the Survived column, as Y.

We can use train_test_split module in python to split our dataset into training and testing.

The python code to be used for this is as follows



```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(X,Y,test_size = 0.3 , random_state = 42)
```

In this way, we will divide our data into 70% training and 30% testing.

The xtrain and ytrain are the datasets we will use for training our model. The xtest and ytest datasets are the datasets we will use to test the accuracy of our model.

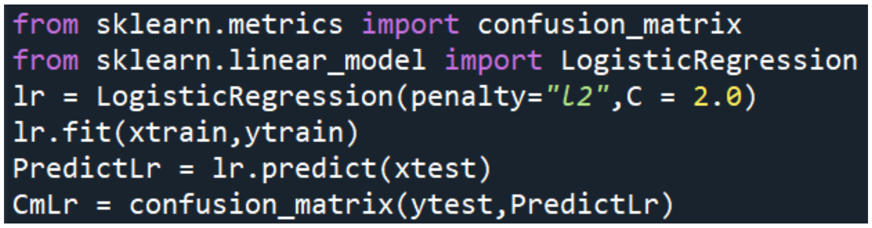
Using classification models and success control

Three classification algorithms were chosen to be used in this project.

- Logistic Regression
- Linear Support Vector Classifier
- Decision Tree Classifier

Classification with Logistic regression

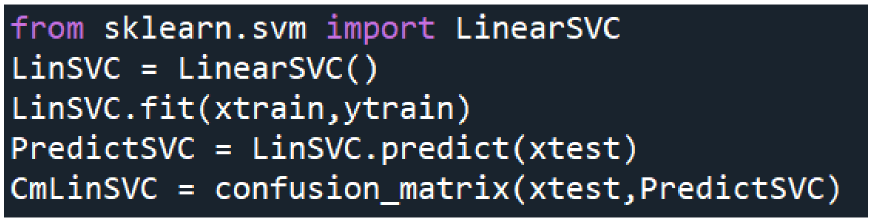
Let's train our model with logistic regression and create our confusion matrix for this model.



```
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(penalty="l2",C = 2.0)
lr.fit(xtrain,ytrain)
PredictLr = lr.predict(xtest)
CmLr = confusion_matrix(ytest,PredictLr)
```

Classification with Linear SVC

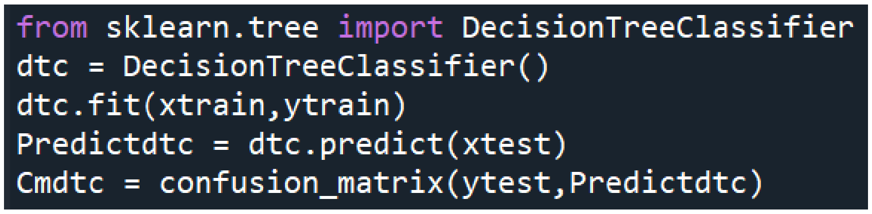
Let's train our model with LinearSVC and create our confusion matrix for this model.



```
from sklearn.svm import LinearSVC
LinSVC = LinearSVC()
LinSVC.fit(xtrain,ytrain)
PredictSVC = LinSVC.predict(xtest)
CmLinSVC = confusion_matrix(xtest,PredictSVC)
```

Classification with Decision Tree Classifier

Let's train our model with Decision Tree Classifier and create our confusion matrix for this model.



```
from sklearn.tree import DecisionTreeClassifier
dtc = DecisionTreeClassifier()
dtc.fit(xtrain,ytrain)
Predictdtc = dtc.predict(xtest)
Cmdtc = confusion_matrix(ytest,Predictdtc)
```

Comparing the success rates of algorithms using error metrics

We used measures of accuracy, balanced accuracy, and ROC/AUC score to compare the success rates of our algorithms. The accuracy metric is a risky value to evaluate our algorithm. Therefore, it would be more beneficial to use a balanced accuracy rate when comparing algorithms. We can use the complexity matrices we have created to see how the numerical values we receive are generated. ***In the results ;***

Accuracy

Logistic Regression : 0.80

SVC : 0.764

Decision Tree Classifier : 0.738

Balanced Accuracy

Logistic Regression : 0.797

SVC : 0.737

Decision Tree Classifier : 0.726

ROC/AUC score

Logistic Regression : 0.797

SVC : 0.568

Decision Tree Classifier : 0.721

As can be seen from the results, the algorithm that gives the most reliable results is the Logistic regression algorithm.

How can the success rate be increased?

Changes can be made to the parameters to change the success rate. For example, changing the criterion parameter (eg entropy) in the decision tree classification algorithm can increase or decrease the success rate.

It can lead to a change in the success rate in the methods to be used in variable selection. For example, using the backward elimination or bidirectional elimination method, eliminating the variables from the data set from bad to good can increase your success rate.
For problems such as overfitting or underfitting, changing the C parameter can increase the success rate.

Mert Aydoğan

Gazi University Department of Mechanical Engineering