Task

This is the continuation of last weeks exercise. After having analysed the Titanic Dataset, you should now prepare a machine learning model to predict whether passengers will survive.

It is entirely up to you which algorithm and feature engineering to use. I do recommend using some of the algorithms available in sklearn, but if you would like to use another library that's also ok. It's a good idea to try and evaluate different algorithms, and different pre-processing/cleaning/feature-generation options if you have the time.

I have split the training data into a train- and a test-set already. These can be found as separate files in the `data` -directory. You should only use the training set throughout your entire development -- feel free to use cross-validation or split the training set into a train- and a validation set again. Once you have developed a final model, you should evaluate this model on the test set I've provided, and report the MCC score for the test set in the title of your PR. You should **not** evaluate the test set more than once for this initial submission!

If you decide to change your code after code-review, you can report new values in the comments, but leave the initial MCC in the title unchanged.

Project

In [ ]:
```
import numpy as np
import pandas as pd
```

Prepare Dateset

In [ ]:
```
df = pd.read_csv('../../data/titanic_train.csv')
df.head(2)
df = df.loc[:, ["Survived", "Pclass", "Sex", "Age", "SibSp", "Parch", "Fare", "E
df["Age"] = df["Age"].fillna(df["Age"].mean())
df["Embarked"] = df["Embarked"].fillna("S")
df["Embarked"] = df["Embarked"].apply(lambda x: {'S':0, 'Q':1, 'C':2}[x])
df["Sex"] = df["Sex"].apply(lambda x: {'female':1, 'male':0}[x])
df.head(2)
```

Out[ ]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	29.567002	0	0	8.05	0
1	1	1	0	51.000000	0	0	26.55	0

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import matthews_corrcoef
x = df.loc[:, df.columns != "Survived"]
```

```
y = df[["Survived"]]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1, stratif
```

Decision Tree Classifier

In [ ]:
```
from sklearn import tree
classifier5 = tree.DecisionTreeClassifier(max_depth=4)
classifier5.fit(x_train, y_train)
matthews_corrcoef(y_test, classifier5.predict(x_test))
```

Out[ ]: 0.45324298151595277

SGD Classifier

In [ ]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(x_train)
x_train_s = scaler.transform(x_train)
x_test_s = scaler.transform(x_test)
```

In [ ]:
```
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV

parms = {
    #'max_iter': [2, 5, 10, 50, 100, 200, 500, 1000]
    "loss": ["hinge", "log_loss", "squared_hinge", "modified_huber", "perceptron
    "alpha": [0.0001, 0.001, 0.01, 0.1],
    "penalty": ["l2", "l1", "elasticnet", None],
}

class1 = SGDClassifier(max_iter=100000)
grid = GridSearchCV(class1, param_grid=parms, cv=10, scoring='matthews_corrcoef'

grid.fit(x_train_s, y_train.values.ravel())
grid.best_params_
```

Out[ ]: {'alpha': 0.001, 'loss': 'log_loss', 'penalty': 'elasticnet'}

In [ ]:
```
from sklearn.model_selection import cross_val_score
for iters in [10, 100, 1000, 5000, 10000, 20000, 100000]:
    class2 = SGDClassifier(max_iter=iters, alpha=0.01, loss='log_loss', penalty=
    #score = cross_val_score(class2, x_train, y_train, scoring='matthews_corrcoe
    class2.fit(x_train_s, y_train.values.ravel())
    print(matthews_corrcoef(y_test, class2.predict(x_test_s)))
```

```
0.4922362680945412
0.523101174836567
0.523101174836567
0.523101174836567
0.5539251650770359
0.523101174836567
0.523101174836567
```

```
c:\Users\Johannes\miniforge3\envs\online-python\Lib\site-packages\sklearn\linear_
model\_stochastic_gradient.py:713: ConvergenceWarning: Maximum number of iteratio
n reached before convergence. Consider increasing max_iter to improve the fit.
  warnings.warn(
```

Choice of parameters:

parameter	choice
max_iter	5000
alpha	0.01
loss	squared_hinge
penalty	l1

Final Model:

In [ ]:
```
classifier = SGDClassifier(max_iter=5000, alpha=0.01, loss='squared_hinge', pena
classifier.fit(x_train_s, y_train.values.ravel())
print(f'Score on training data: {matthews_corrcoef(y_train, classifier.predict(x
print(f'Score on test data: {matthews_corrcoef(y_test, classifier.predict(x_test
```

```
Score on training data: 0.5698426457437183
Score on test data: 0.49816581071166544
```

In [ ]:
```
df = pd.read_csv('../../data/titanic_test.csv')
df.head(2)
df = df.loc[:, ["Survived", "Pclass", "Sex", "Age", "SibSp", "Parch", "Fare", "E
df["Age"] = df["Age"].fillna(df["Age"].mean())
df["Embarked"] = df["Embarked"].fillna("S")
df["Embarked"] = df["Embarked"].apply(lambda x: {'S':0, 'Q':1, 'C':2}[x])
df["Sex"] = df["Sex"].apply(lambda x: {'female':1, 'male':0}[x])

x = df.loc[:, df.columns != "Survived"]
y = df[["Survived"]]


scaler = StandardScaler()
scaler.fit(x)
x_s = scaler.transform(x)
```

In [ ]:
```
print(matthews_corrcoef(y, classifier5.predict(x)))
print(matthews_corrcoef(y, classifier.predict(x_s)))
```

```
0.5482001098230149
0.6477778295184681
```