Examen de Machine Learning

By BAHRI KHALID

In this exam i'm going to use the following libraries and i will explainthe utility in the time i use it

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error as mae
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import LinearSVR
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import VarianceThreshold
```

In order to evaluate the models i build i will use the following two metrices

the MMRE

$$\text{MMRE} = \sum_{i=1}^{n} \frac{\text{MRE}}{\text{N}}$$

OR

$$\text{MMRE} = \frac{\sum_{i=1}^{N} \left| \frac{\text{Actual Effort} - \text{Estimated Effort}}{\text{Actual Effort}} \right| \text{X } 100}{\text{N}}$$

and the Pred(25%)

$$\boldsymbol{Pred(25\%) = \frac{1}{N}\sum_{i=1}^{N} \begin{cases} 1 \; if \; MRE_i \leq 25\% \\ 0 \; otherwise \end{cases}}$$

In the following three cells i will define the metric functions to use

In [2]:

```
def MRE(real,predict):
    return abs((real-predict)/real)
```

In [3]:

```
def MMRE(real,predict):
    sumOfMRE = 0
    m = len(real)
    for i in range(m):
        sumOfMRE+= MRE(real.iloc[i],predict[i])
    return sumOfMRE/m
```

In [4]:

```
def Pred25(real,predict):
    sumOfMRE = 0
    m = len(real)
    for i in range(m):
        sumOfMRE+= 1 if MRE(real.iloc[i],predict[i])<0.25 else 0
    return sumOfMRE/m
```

Now i will use the read_csv function from pandas to read the data and add a new column 'team_size'

In [5]:

```
data = pd.read_csv('./resources/zia.csv',sep = ';')
team_size = np.array([5,5,5,5,5,7,5,5,5,5,5,5,5,7,5,5,5,5,5,5,5])
teamSize = pd.DataFrame(data = {'team_size':team_size})
data["team_size"] = teamSize
data.head()
```

Out[5]:

	storyPoint	velocity	Effort	team_size
0	156	2.7	63	5
1	202	2.5	92	5
2	173	3.3	56	5
3	331	3.8	86	5
4	124	4.2	32	5

In the next cell i will split the data into two parts, the features and the target the i will use the train_test_split function from sklearn.model_selection to split the data into 0.7/0.3 train/test

In [6]:

```
target = data.Effort
data.drop("Effort", axis=1, inplace=True)
# data.drop("velocity", axis=1, inplace=True)
Xtrain_data,Xtest_data,y_train,y_test = train_test_split(data,target,train_
```

In [7]:

```
Xtrain_data.head()
```

Out[7]:

	storyPoint	velocity	team_size
4	124	4.2	5
2	173	3.3	5
6	97	3.4	5
7	257	3.0	5
1	202	2.5	5

In [8]:

```
y_train.head()
```

Out[8]:

```
4    32
2    56
6    35
7    93
1    92
Name: Effort, dtype: int64
```

StandardScaler

**Before make any model i will start by making the data standarized using the SrandarScaler from sklearn.feature_selection

In [9]:

```
mySS = StandardScaler()
Xtrain_data = mySS.fit_transform(Xtrain_data)
Xtest_data = mySS.transform(Xtest_data)
Xtrain_data[:7,:]
```

Out[9]:

```
array([[-0.41174279,  2.5354089 , -0.40824829],
       [ 0.25758316,  0.58747279, -0.40824829],
       [-0.78055504,  0.80391014, -0.40824829],
       [ 1.40499908, -0.06183924, -0.40824829],
       [ 0.65371485, -1.14402597, -0.40824829],
       [-1.09472845, -0.27827659,  2.44948974],
       [ 0.02536804, -0.71115128, -0.40824829]])
```

I will use the VarianceThreshold from sklearn.feature_selection in order to check if i need to remove any variable from the dataset and that using a threshold = 0.8x(1-0.8)

In [10]:

```
sel = VarianceThreshold(threshold=(.8* (1 - .8)))
sel.fit_transform(Xtrain_data)
```

Out[10]:

```
array([[-0.41174279,  2.5354089 , -0.40824829],
       [ 0.25758316,  0.58747279, -0.40824829],
       [-0.78055504,  0.80391014, -0.40824829],
       [ 1.40499908, -0.06183924, -0.40824829],
       [ 0.65371485, -1.14402597, -0.40824829],
       [-1.09472845, -0.27827659,  2.44948974],
       [ 0.02536804, -0.71115128, -0.40824829],
       [-0.86251332, -0.71115128, -0.40824829],
       [-0.23416651, -0.71115128, -0.40824829],
       [ 0.77665227,  0.37103545, -0.40824829],
       [-0.95813132, -1.36046331, -0.40824829],
       [-0.72591619, -0.27827659, -0.40824829],
       [-0.57565935, -0.27827659, -0.40824829],
       [ 2.52509557,  1.23678483,  2.44948974]])
```

You can see that with the threshold of 0.8(1-0.8) all the variables are important so i will keep them all

---

LinearRegression() model

The first model i will make is the LinearRegression from sklearn.linear_model

In [15]:

```
clf = LinearRegression()
clf.fit(Xtrain_data,y_train)
predictedTest = clf.predict(Xtest_data)
predictedTrain = clf.predict(Xtrain_data)
print(f'Train MMRE = {MMRE(y_train,predictedTrain)}')
print(f'Train Pred25 = {Pred25(y_train,predictedTrain)}')
print(f'Test MMRE = {MMRE(y_test,predictedTest)}')
print(f'Test Pred25 = {Pred25(y_test,predictedTest)}')
```

```
Train MMRE = 0.12084195325088669
Train Pred25 = 1.0
Test MMRE = 0.112298930451235
Test Pred25 = 0.8571428571428571
```

RandomForestRegressor model

Now i will make is the RandomForestRegressor from sklearn.ensemble with some specific paraleters

In [ ]:

```
clf2 = RandomForestRegressor(n_estimators=150,verbose=0,max_depth=5,max_lea
clf2.fit(Xtrain_data,y_train)
predictedTest2 = clf2.predict(Xtest_data)
predictedTrain2 = clf2.predict(Xtrain_data)
print(f'Train MMRE = {MMRE(y_train,predictedTrain2)}')
print(f'Train Pred25 = {Pred25(y_train,predictedTrain2)}')
print(f'Test MMRE = {MMRE(y_test,predictedTest2)}')
print(f'Test Pred25 = {Pred25(y_test,predictedTest2)}')
```

LinearSVR model

Finaly i will make is the LinearSVR from sklearn.svm with some specific paraleters

In [14]:

```
clf3  = LinearSVR(max_iter=100000,random_state=0)
clf3.fit(Xtrain_data,y_train)
predicted3 = clf3.predict(Xtest_data)
predictedTrain3 = clf3.predict(Xtrain_data)
print(f'Train MMRE = {MMRE(y_train,predictedTrain3)}')
print(f'Train Pred25 = {Pred25(y_train,predictedTrain3)}')
print(f'Test MMRE = {MMRE(y_test,predicted3)}')
print(f'Test Pred25 = {Pred25(y_test,predicted3)}')
```

```
Train MMRE = 0.6918914064951933
Train Pred25 = 0.0
Test MMRE = 0.7044609137632394
Test Pred25 = 0.0
```

The following plot is using the target variale and the storyPoint and we can see that there is a very strong linear relation between them So that why the first model "LinearRegression" give best resolts in the MMRE while the LinearSVR give bad results

In [ ]:

```
plt.grid()
plt.scatter(data.storyPoint[data.team_size==5],target[data.team_size==5],c=
plt.scatter(data.storyPoint[data.team_size==7],target[data.team_size==7],c=
plt.show()
```

Generale Results

Models	Train_MMRE	Train_PRED25	Test_MMRE	Test_PRED25
LinearRegression	0.12	1	0.11	0.85
RandomForestRegressor	0.06	1	0.16	0.85
LinearSVR	0.69	0	0.7	0