PROGRAMMING IN PYTHON FOR DATA SCIENCE(CS3PP19)
Final Exam: Question 4

52944

April 27, 2021

1 Initialization Code

Load the nessesary libraries + dataset

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn import datasets
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error,r2_score
import numpy as np
from sklearn.metrics import confusion_matrix
import seaborn as sns
data=pd.read_csv("data.csv")
```

standard pre-processing

```
data=data.dropna()
data=data.drop_duplicates()
original=data.copy()
```

and mapping all string category variables into numeric

```
mapping = {
        "school":{"MS":0,"GP":1},
        "sex":{"M":0,"F":1},
        "address":{"U":0,"R":1},
        "famsize":{"LE3":0,"GT3":1},
        "Pstatus":{"T":0,"A":1},
        "Mjob":{"teacher":0,"health":1,"services":2,"at_home":3,"other":4},
        "Fjob":{"teacher":0,"health":1,"services":2,"at_home":3,"other":4},
        "reason":{"home":0,"reputation":1,"course":2,"other":3},
```

```
        "guardian":{"father":0,"mother":1,"other":2},
        "schoolsup":{"yes":0,"no":1},
        "famsup":{"yes":0,"no":1},
        "paid":{"yes":0,"no":1},
        "activities":{"yes":0,"no":1},
        "nursery":{"yes":0,"no":1},
        "higher":{"yes":0,"no":1},
        "internet":{"yes":0,"no":1},
        "romantic":{"yes":0,"no":1}

}
data_numeric=data.replace(mapping)
```

finaly the models

```
train,ttrain,test,ttest=train_test_split(data_numeric,target,test_size=0.2,random_sta
model1=LogisticRegression(max_iter=500000)
model1.fit(train,test)
model1.score(ttrain,ttest)
model2=svm.SVC(gamma=0.001, C=100.)
model2.fit(train,test)
model2.score(ttrain,ttest)
```

2 Q4.a.i

We will perform a kfold cross validation, with k=10. First for Logistic regression

```
k=10
model1scores=[]
for i in range(k):
        train,ttrain,test,ttest=train_test_split(data_numeric,target,test_size=0.2,ra
        model1=LogisticRegression(max_iter=50000)
        model1.fit(train,test)
        model1scores.append(model1.score(ttrain,ttest))
```

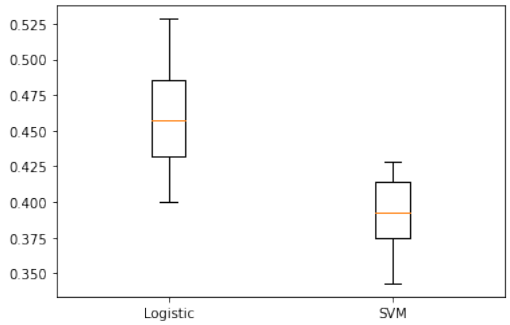
And then for Support Vector Machine

```
k=10
model2scores=[]
for i in range(k):
        train,ttrain,test,ttest=train_test_split(data_numeric,target,test_size=0.2,ra
        model2=svm.SVC(gamma=0.001, C=100.)
        model2.fit(train,test)
        model2scores.append(model2.score(ttrain,ttest))
```

Finally we will produce a boxplot of the variance of the scores, to compare

```
kfold=pd.DataFrame()
kfold['Logistic']=pd.Series(model1scores)
kfold['SVM']=pd.Series(model2scores)
plt.boxplot(kfold,labels=['Logistic','SVM'])
```

Figure 1: Class Counts on grades



Becomes evident that the Logistic Regression performs slighly better than the SVC, this is something to be expected though, due to the nature of the algorithms themselfes. SVC is a pure geometric algorithm that tries to 'fit' a prediction to the nearest cluster, Logistic Regression in contrast, uses the statistical properties of the datapoints to predict an unknown datapoint. This by itself is not an issue, but if we take into account the fact that this dataset has inbalanced clusters, this makes sence. fewer datapoints on specific grades, bigger the probability of a wrong classification for SVC. By running the following command, this becomes evident

```
original.groupby('grade').grade.count()
```

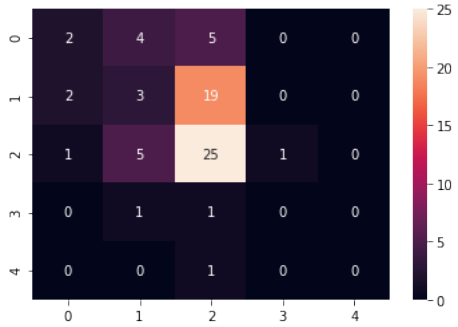
A	39
B	91
C	189
D	22
E	7

3 Q4.a.ii

Using the skmetrics method confusion_matrix as well as the seaborns heatmap plot, we can visualize the matrix in a nice way. Lets apply on LogisticRegression

```
train,ttrain,test,ttest=train_test_split(data_numeric,target,test_size=0.2,random_sta
model1=LogisticRegression(max_iter=500000)
model1.fit(train,test)
cf_matrix = confusion_matrix(ttest, model1.predict(ttrain))
sns.heatmap(cf_matrix,annot=True)
```

Figure 2: Confusion-matrix-heatmap:LogisticRegression



And on SVM as well...

```
train,ttrain,test,ttest=train_test_split(data_numeric,target,test_size=0.2,random_sta
model2=svm.SVC(gamma=0.001, C=100.)
model2.fit(train,test)
cf_matrix = confusion_matrix(ttest, model2.predict(ttrain))
sns.heatmap(cf_matrix,annot=True)
```

Figure 3: Confusion-matrix-heatmap:SVM