Lab7

August 28, 2020

1 Import Libraries

[65]:
```
import numpy as np
import sklearn
import pandas as pd
from sklearn.datasets import load_breast_cancer
import matplotlib.pyplot as plt
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
import seaborn as sns
from sklearn.linear_model import LogisticRegression as LR
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.neighbors import KNeighborsClassifier as KNN
```

2 Load Breast Cancer Dataset

[2]:
```
br=load_breast_cancer()
data=np.c_[br.data,br.target]
columns=np.append(br.feature_names, ["target"])
df=pd.DataFrame(data, columns=columns)
df
```

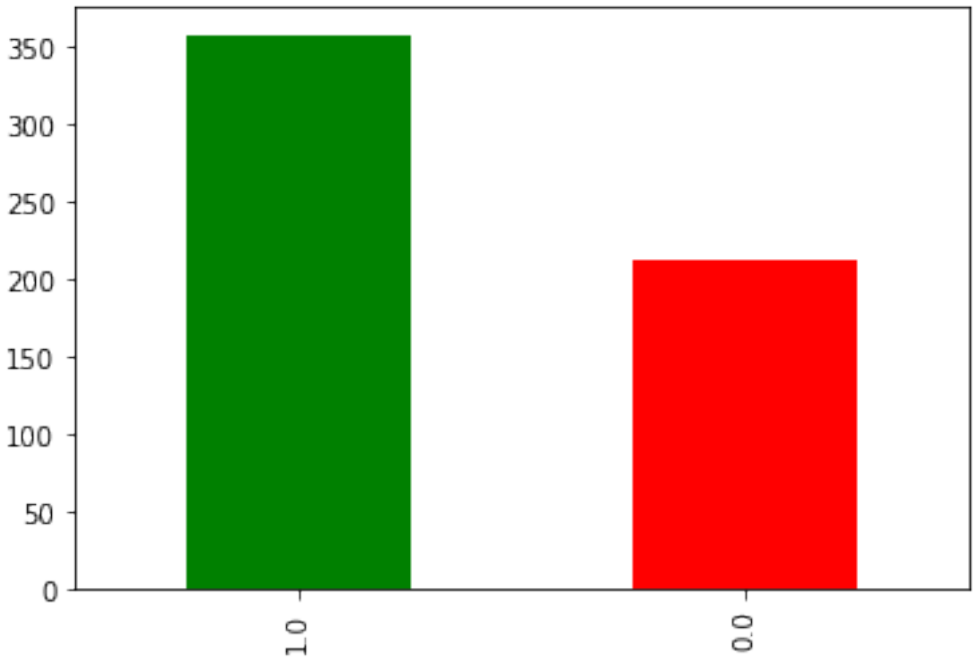
[2]:

	mean radius	mean texture	...	worst fractal dimension	target
0	17.99	10.38	...	0.11890	0.0
1	20.57	17.77	...	0.08902	0.0
2	19.69	21.25	...	0.08758	0.0
3	11.42	20.38	...	0.17300	0.0
4	20.29	14.34	...	0.07678	0.0
..	...	...	...	...	...
564	21.56	22.39	...	0.07115	0.0
565	20.13	28.25	...	0.06637	0.0
566	16.60	28.08	...	0.07820	0.0
567	20.60	29.33	...	0.12400	0.0
568	7.76	24.54	...	0.07039	1.0

```
[569 rows x 31 columns]
```

[12]:
```
df['target'].value_counts().
 ↪plot(kind='bar',y=['benign','malignant'],color=['green','red'])
v=df['target'].value_counts().to_dict()
print("Benign tumour counts:"+str(v[1.0]))
print("Malignant tumour counts:"+str(v[0.0]))
```

```
Benign tumour counts:357
Malignant tumour counts:212
```



[26]:
```
label_dict = {0:'Benign', 1: 'Malignant', 2: 'Class 3', 3:'Class 4'}
plt.figure(figsize=(7,7))

def plot_scikit_lda(X,y, title):
      ax = plt.subplot(111)
      for label,marker,color in zip(
          [i for i in␣
 ↪range(2)],('s','^','o','*'),('green','red','blue','black')):

          plt.scatter(x=X[:,0][y == label],
                      y=X[:,1][y == label],
                      marker=marker,
                      color=color,
                      alpha=0.5,
                      label=label_dict[label])
```

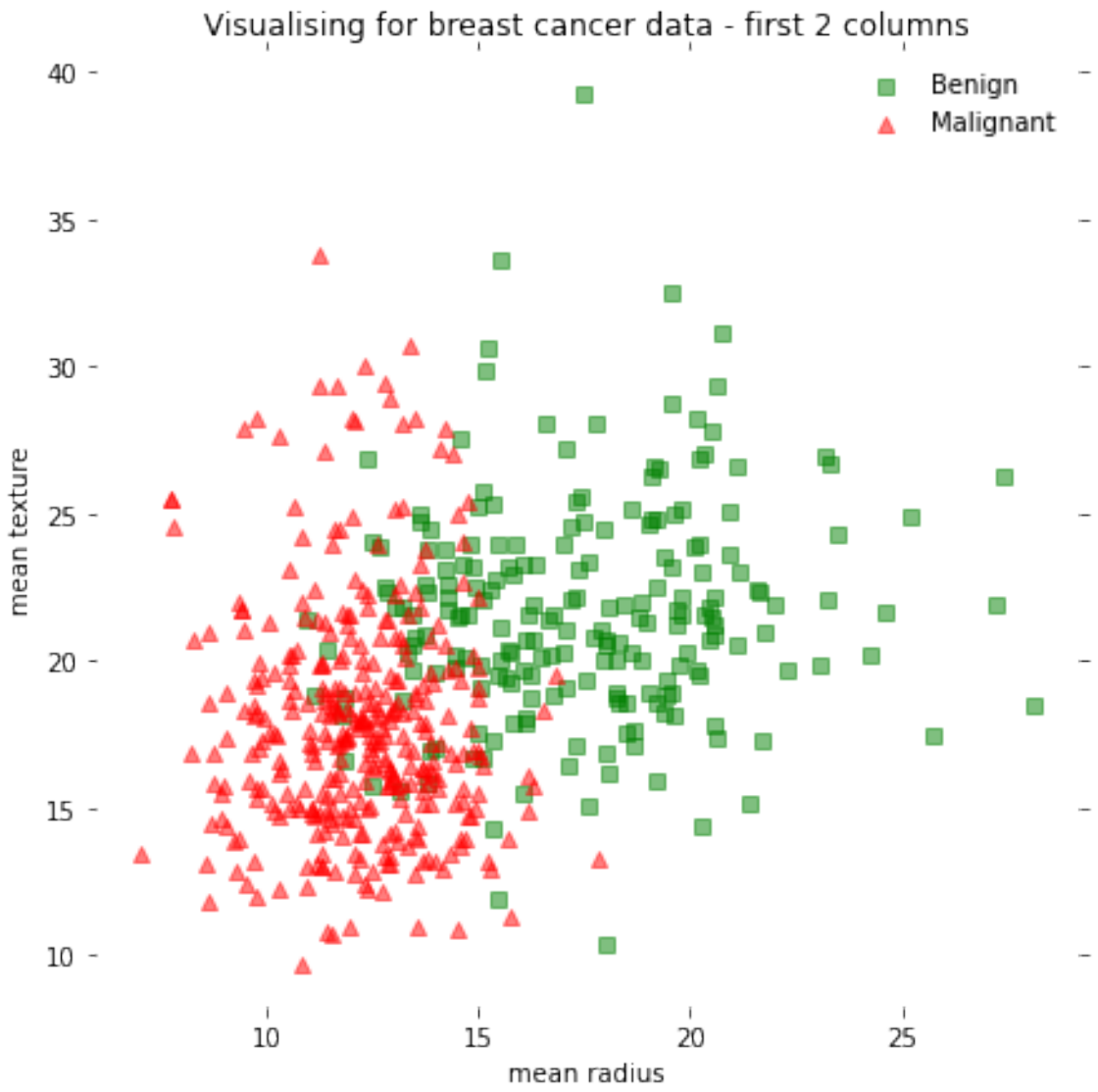
```
        plt.xlabel(df.columns[0])
        plt.ylabel(df.columns[1])

        leg = plt.legend(loc='upper right', fancybox=True)
        leg.get_frame().set_alpha(0)
        plt.title(title)

        # hide axis ticks
        plt.tick_params(axis="both", which="both", bottom="off", top="off",
                labelbottom="on", left="off", right="off", labelleft="on")

        # remove axis spines
        ax.spines["top"].set_visible(False)
        ax.spines["right"].set_visible(False)
        ax.spines["bottom"].set_visible(False)
        ax.spines["left"].set_visible(False)
        plt.tight_layout
        plt.show()

plot_scikit_lda(np.array(df.iloc[:,:2]), np.array(df.iloc[:,-1]),␣
 ↪title='Visualising for breast cancer data - first 2 columns')
```



[31]:
```
x_train,x_test,y_train,y_test=train_test_split(df.iloc[:,:df.shape[1]-1],df.
↪iloc[:,-1],random_state=0)
```

3 Logistic Regression without penalty

[59]:
```
lr1=LR(max_iter=10000,penalty='none')
l1=lr1.fit(x_train,y_train)
y_pred1=l1.predict(x_test)
print('Accuracy:',l1.score(x_test,y_test)*100)
print(classification_report(y_true=y_test,y_pred=y_pred))
```

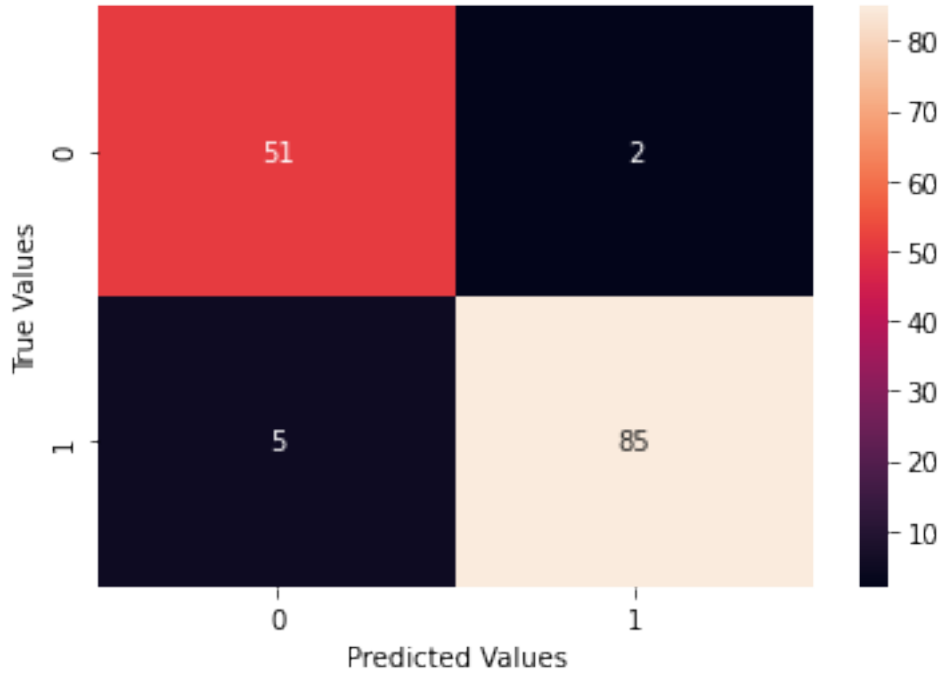
```
Accuracy: 95.1048951048951
```

```
              precision    recall  f1-score   support

         0.0       0.91      0.98      0.95        53
         1.0       0.99      0.94      0.97        90

    accuracy                           0.96       143
   macro avg       0.95      0.96      0.96       143
weighted avg       0.96      0.96      0.96       143
```

[60]:
```
sns.heatmap(confusion_matrix(y_test,y_pred1),annot=True)
plt.xlabel('Predicted Values')
plt.ylabel('True Values')
plt.show()
```



4 Logistic Regression with L2 or ridge penalty

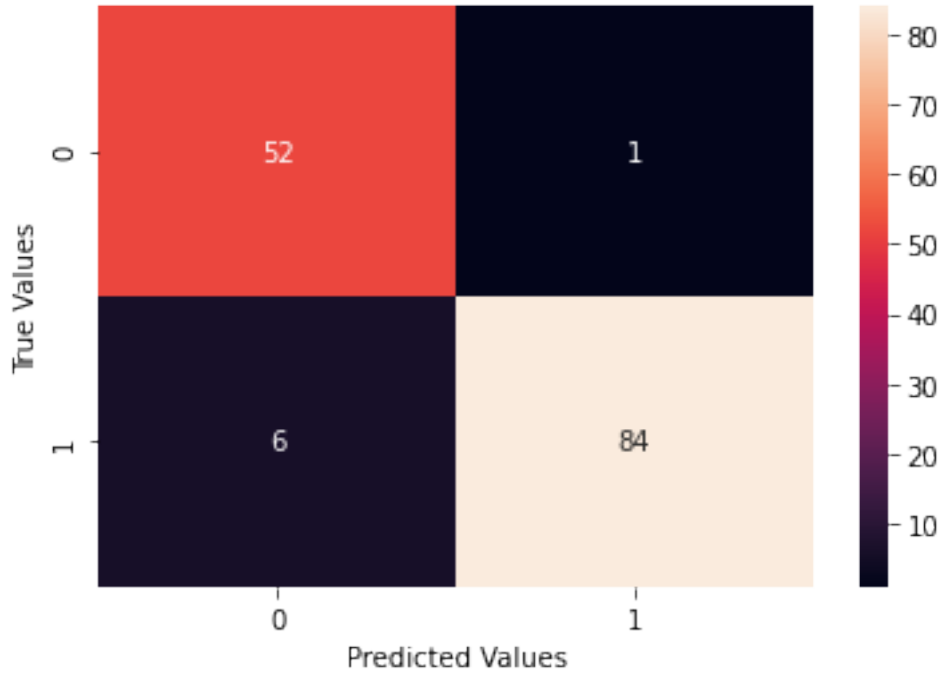
[61]:
```
lr2=LR(max_iter=10000,penalty='l2')
l2=lr2.fit(x_train,y_train)
y_pred2=l2.predict(x_test)
print('Accuracy:',l2.score(x_test,y_test)*100)
print(classification_report(y_true=y_test,y_pred=y_pred))
```

```
Accuracy: 95.1048951048951
              precision    recall  f1-score   support

         0.0       0.91      0.98      0.95        53
         1.0       0.99      0.94      0.97        90

    accuracy                           0.96       143
   macro avg       0.95      0.96      0.96       143
weighted avg       0.96      0.96      0.96       143
```

[62]:
```
sns.heatmap(confusion_matrix(y_test,y_pred2),annot=True)
plt.xlabel('Predicted Values')
plt.ylabel('True Values')
plt.show()
```



5 Logistic Regression with L1 or lasso penalty

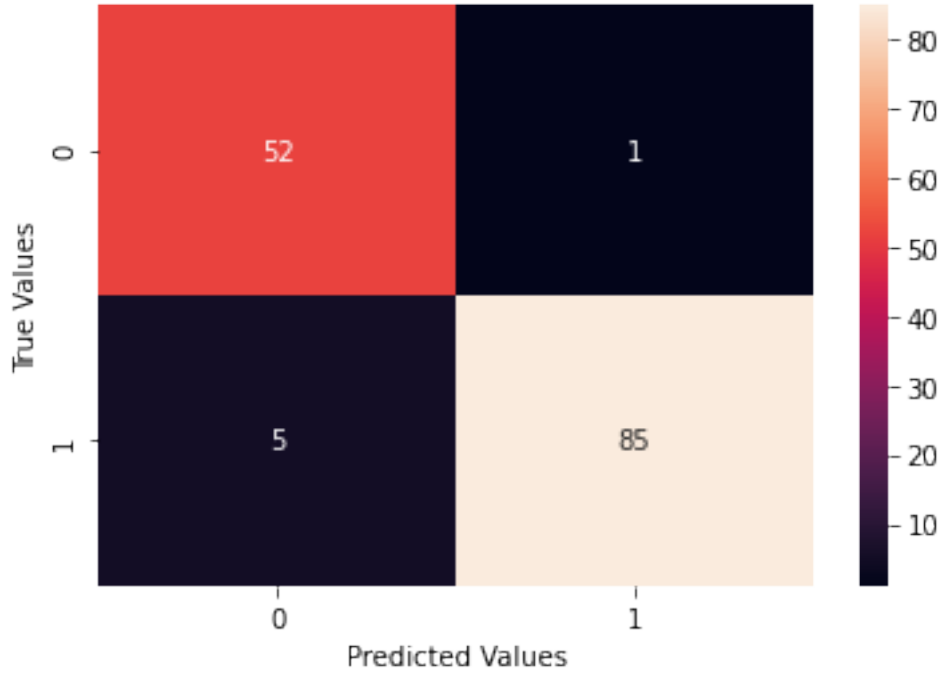
[63]:
```
lr3=LR(max_iter=1000,penalty='l1',solver='liblinear')
l3=lr3.fit(x_train,y_train)
y_pred3=l3.predict(x_test)
print('Accuracy:',l3.score(x_test,y_test)*100)
print(classification_report(y_true=y_test,y_pred=y_pred))
```

```
Accuracy: 95.8041958041958
              precision    recall  f1-score   support

         0.0       0.91      0.98      0.95        53
         1.0       0.99      0.94      0.97        90

    accuracy                           0.96       143
   macro avg       0.95      0.96      0.96       143
weighted avg       0.96      0.96      0.96       143
```

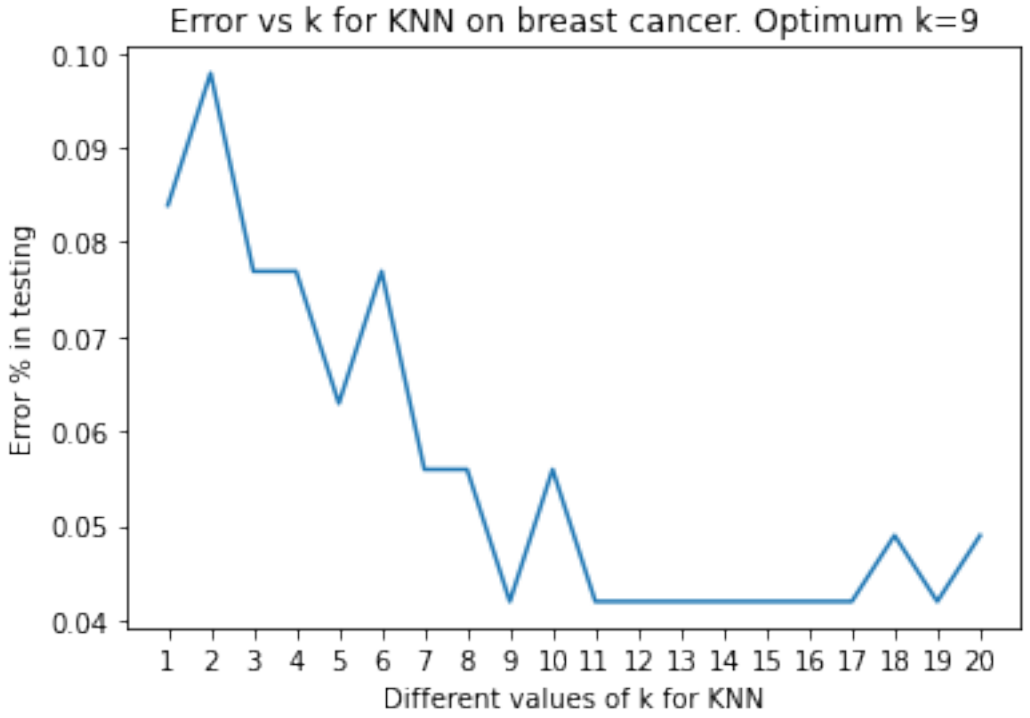
[64]:
```
sns.heatmap(confusion_matrix(y_test,y_pred3),annot=True)
plt.xlabel('Predicted Values')
plt.ylabel('True Values')
plt.show()
```



6 KNN Classifier

[94]:
```
errs=[]
for i in range(1,21):
  k=KNN(n_neighbors=i)
  kn=k.fit(x_train,y_train)
  acc=kn.score(x_test,y_test)
  errs.append(1-acc)
```

```
plt.plot([i for i in range(1,21)],errs)
plt.xlabel('Different values of k for KNN')
plt.xticks([i for i in range(1,21)])
plt.ylabel('Error % in testing')
plt.title('Error vs k for KNN on breast cancer. Optimum k='+str(errs.index(np.
 ↪min(errs))+1))
plt.show()
min_k=errs.index(np.min(errs))
```



[98]:
```
k=KNN(n_neighbors=min_k+1)
kn=k.fit(x_train,y_train)
k_pred=kn.predict(x_test)
print("Accuracy of KNN clasifier")
print(classification_report(y_true=y_test,y_pred=k_pred))
```

```
Accuracy of KNN clasifier
              precision    recall  f1-score   support

         0.0       0.96      0.92      0.94        53
         1.0       0.96      0.98      0.97        90

    accuracy                           0.96       143
```

```
   macro avg       0.96      0.95      0.95       143
weighted avg       0.96      0.96      0.96       143
```

[99]:
```
sns.heatmap(confusion_matrix(y_test,k_pred),annot=True)
plt.xlabel('Predicted Values')
plt.ylabel('True Values')
plt.show()
```