Questions

a) For the code in the colab file we sent (from the baseline model) – how would you add logistic regression to the model? (in code i.e. your output should be code only)

Answer

```
# import main data analysis libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
# note we use scipy for generating a uniform distribution in the model optimization step
from scipy.stats import uniform

# note that because of the different dataset and algorithms, we use different sklearn libraries from Day 1
from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

# hide warnings
import warnings
warnings.filterwarnings('ignore')

# we load the dataset and save it as the variable data
data = load_breast_cancer()
X1 = pd.DataFrame(data1.data, columns = data1.feature_names)
X1.head()
#print target variable
y1 = data1.target
y1
#print dataframe columns
X1.columns
#create train/test split and initiate the model and fit it
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size = 0.30, random_state = 0)
clf_reg = LogisticRegression(random_state = 0)
clf_reg.fit(X1_train, y1_train)
#predict the model on test data
y1_pred = clf_reg.predict(X1_test)
#find the classifier accuracy
cm = confusion_matrix(y1_test, y1_pred)
cm
#get the accuracy score
accuracy_score(y1_test, y1_pred)

#to get the count instead of ratio
```

```
accuracy_score(y1_test, y1_pred, normalize=False)
```

b) For the code in the colab file we sent (from the baseline model) – how would you add linear regression to the model? (just list the steps – not the code for this)

Answer

Steps are below

1-Load the required libraries

2-Load the dataset

3-Perform pre-processing

- ➔ Missing value
- ➔ Treat categorical values etc

4-Perform Exploratory Data Analysis

5- plot the data to see the correlation and make your understanding more better

6- Perform train/test split.

7- Choose baseline algorithm-linearRegression using Sklearn lib

8-Train/test your model and define the split

9-Predict on test data from the split above.

10-choose evaluation metric to find the accuracy of the model- Mean Absolute Error or Means Squared Error

11-improve dataset, in case of any scaling/normalization that could help to make the result better.

12-test alternative models as well.

11-choose the best model and tune the hyper parameter to get the best score for your model-using RandomSearchCV.

c) For KNN algorithm in the code, how did we choose these values for the hyperparameter stage
tuned_parameters = {&#39;n_neighbors&#39;: [2, 4, 6, 8, 10]}?
Answer

n_neighbors represents the number of neighbors to use for k-neighbors queries
Using n_neighbors=1 means each sample is using itself as reference, that's an overfitting case. For getting best score, increasing the number of neighbors improves the test scores
K in KNN is a hyperparameter and hence must select values in order to get the best possible fit for the data set. K value can also help to control the shape of the decision boundary.
When K is small, we are restraining the region for given prediction and forcing our classifier to be “more blind” to the overall distribution. A small value for K provides the most flexible fit, which will have low bias but high variance. On the other hand, a higher K averages more values in each prediction and hence is more resilient to outliers. Larger values of K will have smoother decision boundaries which means lower variance but increased bias.

d) In y = mx + c (equation of a straight line for Ordinary Linear Regression) , which are the hyperparameters?

Answer

M and c are hyper parameters, which we can control and select values to get the best fit for the line.