20 JUNE 2021

MACHINE LEARNING PROJECT REPORT

GROUP NAME: PIRASA2

DATASET: https://www.kaggle.com/brjapon/gearbox-fault-diagnosis

Student1 Name: Muhammed Furkan Yılmaz	Student1 No: 181805069
Student2 Name: Ferhat Kamalı	Student2 No: 171805056

QUESTION 1; Choosing a data set

In this stage, we tried hard to find our own data.

- First we have written a program to send request to a website and extract that particular page into pieces, so that data can be gathered. But the program was buggy and it was not a fast way to collect the data, so we gave up.

```python
import bs4
import requests

title = []
attributes = []
prices = []
dates = []

model = []
kilometers = []
color = []

def getData(soup):
    print("a")
    title_data = soup.find_all('a', class_="classifiedTitle")
    attributes_data = soup.find_all('td', class_="searchResultsAttributeValue")
    prices_data = soup.find_all('td', class_="searchResultsPriceValue")
    dates_data = soup.find_all('td', class_="searchResultsDateValue")


    for data in title_data:
        title.append(data.text)

    for data in attributes_data:
        attributes.append(data.text)

    for data in prices_data:
        prices.append(data.text)

    for data in dates_data:
        dates.append(data.text)

    for i in range(1,len(attributes),3):
        kilometers.append(attributes[i])
        model.append(attributes[i-1])
        color.append(attributes[i+1])


response = requests.get("https://www.sahibinden.com/volkswagen-polo-1.4/benzin-lpg/manuel?a5_max=2010&addressTitle=Adres+Bilgileri&a8=62068&a5_min=2000&price_min=90000&a9213=136269&price_max=
soup = bs4.BeautifulSoup(response.text, 'lxml')

print(soup)

getData(soup)
```

- Secondly, we made an API request to twitter. We applied for twitter developer account and after emailing, proving our project to the managers, we are accepted to reach twitter API.

But the information we could gather from twitter was not compatible with machine learning algorithm, because neither number of likes nor number of retweets had any constant connection with each other.

```python
# def takeTheNames(results):
#     for result in results:
#         if result.source == "Twitter for Android" or result.source == "Twitter for iPhone":
#             if result.user.screen_name not in Names:
#                 Names.append(result.user.screen_name)
#                 followers.append(result.user.followers_count)

# #belirli bir tagdan twit çekmek
# results = api.search(q = "#pazartesi", lang = "tr", result_type="mixed", count = 100)
# takeTheNames(results)
# results = api.search(q = "#bitcoin", lang = "tr", result_type="mixed", count = 100)
# takeTheNames(results)
# results = api.search(q = "#SpaceX", lang = "tr", result_type="mixed", count = 100)
# takeTheNames(results)
# results = api.search(q = "#Lakers", lang = "tr", result_type="mixed", count = 100)
# takeTheNames(results)
# results = api.search(q = "#Beşiktaş", lang = "tr", result_type="mixed", count = 100)
# takeTheNames(results)

def takeFavs(Name):
    tweets = api.user_timeline(id=Name,count = 100)
    favCount = 0
    reTCount = 0

    for tweet in tweets:
        if tweet.text[0] != "R" and tweet.text[1] != "T":
            favCount += tweet.favorite_count
            reTCount += tweet.retweet_count

    sumFavs.append(favCount)
    sumReTs.append(reTCount)
```

We mailed and informed you (course teacher) about this and you told us to choose one of your given data sets for our project.

So we choose the third data set "Gearbox Fault Diagnosis" and continued our project there.

Since you told there can be only 2 members of a group if one of your data sets is taken, we had to divide our group and Anıl friend had to do himself. This is why group's name is different than the first homework.

QUESTION 2: Preprocessing and feature set extraction

The data set came in many pieces that are from different sources but thankfully in the comment section of the data set, there was a little code to mix all these data sets into two pieces, that are divided by their results. https://www.kaggle.com/lucasbr8/gearbox-fault-aggregated-dataset

We used that code to mix the data sets, but this was also not enough since we needed only one file to work with, so we decided to merge those two files into one csv file.

```
# #Combine the two databases we have into one single file

# all_files = glob.glob(os.path.join('results/', "*.csv"))
# df_from_each_file = (pd.read_csv(f, sep=',') for f in all_files)
# df_merged   = pd.concat(df_from_each_file, ignore_index=True)
# #Save the one dataframe table
# df_merged.to_csv( "merged.csv")
```

Then we loaded the "merged.csv" data set and our data is ready.

In the preprocessing part, we used "Local Outlier Factor" library to determine extreme values, then we removed the values so they won't affect our learning

```
#fitting data to preprocessing function
clf = LocalOutlierFactor(n_neighbors=3, contamination=0.1)
clf.fit_predict(preprocessed)

#taking the results according to the function
scores = clf.negative_outlier_factor_

#choosing a threshold value
threshold = np.sort(scores)[3]

#spotting the values that are out of threshold
outlier_tf = scores > threshold

#show values that are out of threshold
#preprocessed[scores < threshold].head

#removing the spotted values
preprocessed = preprocessed[scores > threshold]
```

After the preprocessing, we extracted train and test values according to instructions in project paper

```
#Train Test Splitting
X,y = divideDependent(preprocessed)
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)

#Check if 0 and 1 values are spread evenly
y_train["failure"].value_counts()
y_test["failure"].value_counts()
#We can see that values equal
```

```
def divideDependent(df):
    #Dependent variable
    X = df.drop("load", axis=1)
    X = X.drop("failure", axis=1)
    #Independent Variable
    y = df[["failure"]]

    return X,y
```

QUESTION 3: Learning with different regression models

In this stage, we repeatedly trained all the regression algorithms wanted, tested the accuracy and saved the cross validation scores into a "crossScores" list variable.

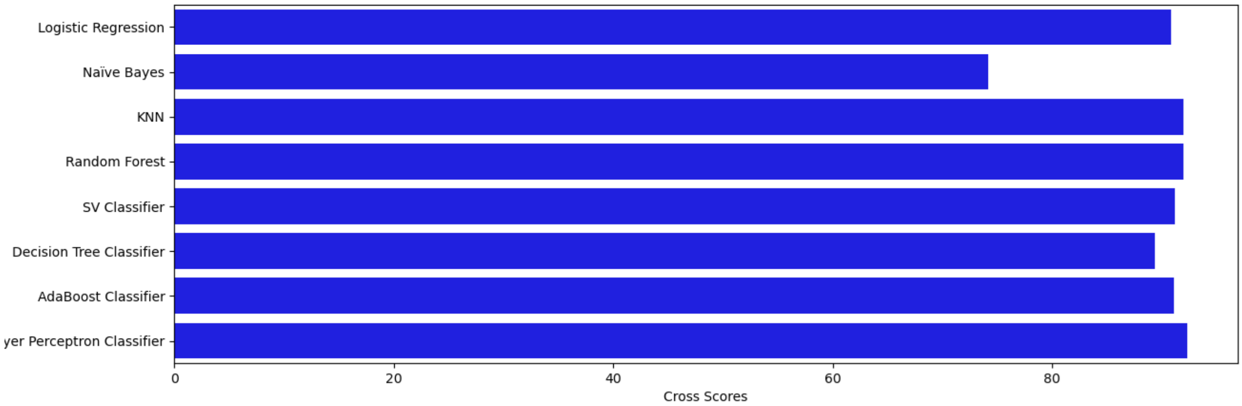
```
############################################### Logistic Regression
#Train the model
loj = LogisticRegression(solver="liblinear")
loj_model = loj.fit(x_train,y_train)
#Test the model
y_pred = loj_model.predict(x_test)
accuracy_score(y_test, y_pred)

#Save the Cross validation score
crossScores.append(cross_val_score(loj_model, x_test, y_test, cv=10).mean())

############################################### Naïve Bayes
#Train the model
nb = GaussianNB()
nb_model = nb.fit(x_train,y_train)
#Test the model
y_pred = nb_model.predict(x_test)
accuracy_score(y_test, y_pred)

#Save the Cross validation score
crossScores.append(cross_val_score(nb_model, x_test, y_test, cv=10).mean())

############################################### KNN
#Train the model
knn = KNeighborsClassifier()
knn_model = knn.fit(x_train,y_train)
#Test the model
y_pred = knn_model.predict(x_test)
accuracy_score(y_test, y_pred)

#Save the Cross validation score
crossScores.append(cross_val_score(knn_model, x_test, y_test, cv=10).mean())
```

Here is all the regression algorithms that are used.

```
from sklearn.neighbors import LocalOutlierFactor,KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict,GridSearchCV
from sklearn.metrics import mean_squared_error,accuracy_score
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier,AdaBoostClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
```

QUESTION 4: Compare cross validation scores and plot

We have already calculated and saved the crossScores in a list in question 3. In this stage we are only plotting the results.



QUESTION 5: Hyperparameter with the best algorithm

In question 4, we seen that the best resulting algorithm is Multi-Layer Perceptron Classifier algorithm.
In order to find the parameters of this algorithm, we used "?mlp_model" code and checked the parameters section.

```
Parameters
----------
hidden_layer_sizes : tuple, length = n_layers - 2, default=(100,)
    The ith element represents the number of neurons in the ith
    hidden layer.

activation : {'identity', 'logistic', 'tanh', 'relu'}, default='relu'
    Activation function for the hidden layer.

    - 'identity', no-op activation, useful to implement linear bottleneck,
      returns f(x) = x

    - 'logistic', the logistic sigmoid function,
      returns f(x) = 1 / (1 + exp(-x)).

    - 'tanh', the hyperbolic tan function,
      returns f(x) = tanh(x).

    - 'relu', the rectified linear unit function,
      returns f(x) = max(0, x)

solver : {'lbfgs', 'sgd', 'adam'}, default='adam'
    The solver for weight optimization.
```

Since in the question it is asked to try 10 different values for each parameter. We had to choose parameters with integer or float values, because other parameters had only few amount of inputs.

So we choose the "Hidden layer sizes", "Alpha", "Learning Rate" parameters to change and see the results.

```
hiddenLayerParam = {"hidden_layer_sizes": [100,150,200,250,300,350,400,450,500,550]}
alphaParam = {"alpha": [0.0001,0.0002,0.0003,0.0004,0.0005,0.0006,0.0007,0.0008,0.0009,0.0010]}
learnRateParam = {"learning_rate_init": [0.001,0.002,0.003,0.004,0.005,0.006,0.007,0.008,0.009,0.010]}
HparamAccuracy = []
bestHparam = []

#Apply the first Hyperparameter
mlp = MLPClassifier()
mlp_cv = GridSearchCV(mlp, hiddenLayerParam, cv=10)
mlp_cv.fit(x_train, y_train)
bestHparam.append(mlp_cv.best_params_)

#Test the hyperparameter accuracy and save it
mlp_paramed1 = MLPClassifier(hidden_layer_sizes = bestHparam[0]["hidden_layer_sizes"])
mlp_paramed1.fit(x_train, y_train)

y_pred = mlp_paramed1.predict(x_test)
HparamAccuracy.append(accuracy_score(y_test, y_pred))

#Apply the second Hyperparameter
```

We written 10 random values for each parameter and tried them one by one to see the results.

Best values for each parameter are:

```
[{'hidden_layer_sizes': 550}, {'alpha': 0.0007},
{'learning_rate_init': 0.003}]
```

Accuracy level for those values are:

```
[0.925, 0.9252475247524753, 0.9237623762376238]
```

QUESTION 6: Use all three parameters as combination

We used all three parameters’ best values as combination to see the last accuracy level.

```
#Use all 3 best parameters as combination
mlp_paramedComb = MLPClassifier(hidden_layer_sizes = bestHparam[0]["hidden_layer_sizes"], alpha = bestH
mlp_paramedComb.fit(x_train, y_train)

y_pred = mlp_paramedComb.predict(x_test)
HparamCombAcc = accuracy_score(y_test, y_pred)

print(HparamCombAcc)
```

But the result was interestingly, not as high as one of the parameter used...

```
0.9207920792079208
```

QUESTION 7: Predicting random inputs

In this last question, we used Random library to gather random inputs and predicted results according to these random inputs. (Result is different in each run, since random numbers change every time)
(Random number limits are chosen according to the actual value limits.)

```
#predict results from random inputs
a1, a2, a3, a4 = [],[],[],[]

for i in range(10):
    a1.append(random.uniform(1, 20))
    a2.append(random.uniform(1, 10))
    a3.append(random.uniform(1, 10))
    a4.append(random.uniform(1, 10))

x_rand = {"a1":a1, "a2":a2, "a3":a3, "a4":a4}
x_rand = pd.DataFrame(x_rand, columns = ["a1","a2","a3","a4"])
y_rand_pred = mlp_paramedComb.predict(x_rand)
#add predicted values to the dataframe
x_rand["results"] = y_rand_pred
```

So the result random data set is like:

Index	a1	a2	a3	a4	results
0	1.1816	4.33561	4.12575	7.33367	1
1	1.78025	1.82962	5.00829	2.71574	1
2	14.9327	9.67403	9.84141	7.97927	0
3	7.62638	3.67457	5.90723	8.59765	0
4	15.6424	4.07972	9.14671	4.26639	0
5	18.6393	1.90971	3.80114	5.2129	0
6	4.28903	1.97925	3.39617	2.7019	0
7	6.02475	9.40972	3.26711	2.55982	0
8	17.6922	5.65501	4.67078	3.45896	0
9	13.907	4.66035	4.88037	4.18653	0

CONCLUSION

The highest accuracy level we ever got with this data set is 0.9252475247524753.
This accuracy level is reached by Multi-Layer Perceptron Classifier regression model, that has “alpha” parameter set to value of 0.0007.