Homework 3

1. Download from the Kaggle site

```
import re
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.svm import LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import preprocessing
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt
import os
```

```
#def paren_match(page, text):
def paren_match(row):
    page = row['page']
    text = row['text']
    page = re.sub('[\(|.|\)|,|"|\t]', '', page)
    page = page.split(' ')
    page = [x.lower() for x in page]
    text = re.sub('[\(|.|\)|,|"|\t]', ' ', text)
    text = text.split(' ')
    text = [x.lower() for x in text]
    count = 0
    for item in page:
        if item in text and item not in [' ', 'at', 'a', 'of', 'the', 'on']:
            count += 1
    return count
```

```
cd /content/drive/MyDrive/808W Assignments/HW3
```

```
/content/drive/MyDrive/808W Assignments/HW3
```

```
train = pd.read_csv('qb.train.csv', sep=',', header=0)
test = pd.read_csv('qb.test.csv', sep=',', header=0)
```

```
print('Columns in train data: ', train.columns.values)
```

```
Columns in train data:  ['row' 'body_score' 'page' 'answer' 'text' 'category' 'tournaments'
 'answer_type' 'corr' 'inlinks']
```

```
print(train.shape[0])
```

```
8079
```

```
print('columns in test data: ', test.columns.values)
```

```
columns in test data:  ['row' 'body_score' 'page' 'text' 'category' 'tournaments' 'answer_type'
 'inlinks']
```

```
print(train.head())
```

```
   row  body_score                 page               answer  \
0    1  127.398036  Comus (John Milton)  Comus (John Milton)
1    2   50.212336                Circe  Comus (John Milton)
2    3   44.767071                Satyr  Comus (John Milton)
3    4   44.058274       Philip K. Dick         Wilfred Owen
4    5   40.675249     Honore de Balzac         Wilfred Owen

                                                text    category  \
0  First performed in Ludlow Castle by the childr...  Literature
1  First performed in Ludlow Castle by the childr...  Literature
2  First performed in Ludlow Castle by the childr...  Literature
3  This author is convinced by another to publish...  Literature
4  This author is convinced by another to publish...  Literature

          tournaments answer_type   corr  inlinks
0  2000 ACF Nationals        work   True       62
1  2000 ACF Nationals        None  False        5
2  2000 ACF Nationals        None  False        6
3     2009 ACF Winter      people  False       22
4     2009 ACF Winter        None  False        0
```

```
train['paren_match'] = train.apply(paren_match, axis=1)
train['obs_len'] = train['text'].apply(len)
```

```
scaler = MinMaxScaler()
train['inlinks'] += 1
train['inlinks_log'] = np.log2(train['inlinks'])
train[['inlinks_scaled']]  = scaler.fit_transform(train[['inlinks_log']])
bs=[train['body_score']]
bsnorm =  preprocessing.normalize(bs)
bsnorm = bsnorm[0]*100
train['bsnorm'] =  bsnorm
print(train.head())
#print(bsnorm)
```

```
   row  body_score                 page               answer  \
0    1  127.398036  Comus (John Milton)  Comus (John Milton)
1    2   50.212336                Circe  Comus (John Milton)
2    3   44.767071                Satyr  Comus (John Milton)
3    4   44.058274       Philip K. Dick         Wilfred Owen
4    5   40.675249     Honore de Balzac         Wilfred Owen

                                                text    category  \
0  First performed in Ludlow Castle by the childr...  Literature
1  First performed in Ludlow Castle by the childr...  Literature
2  First performed in Ludlow Castle by the childr...  Literature
3  This author is convinced by another to publish...  Literature
```

```
4  This author is convinced by another to publish...  Literature

          tournaments answer_type   corr  inlinks  paren_match  obs_len  \
0  2000 ACF Nationals        work   True       63            2      352
1  2000 ACF Nationals        None  False        6            1      352
2  2000 ACF Nationals        None  False        7            0      352
3     2009 ACF Winter      people  False       23            0      327
4     2009 ACF Winter        None  False        1            0      327

   inlinks_log  inlinks_scaled    bsnorm
0     5.977280        0.388181  1.994746
1     2.584963        0.167875  0.786204
2     2.807355        0.182317  0.700944
3     4.523562        0.293773  0.689846
4     0.000000        0.000000  0.636876
```

```
train = pd.get_dummies(train, columns = ['category', 'tournaments', 'answer_type', \
'corr'])
```

```
features = pd.DataFrame(train, columns=['body_score', 'inlinks'])#,'bsnorm'])
```

```
y = train['corr_True']
#x = train.drop(['corr_True'], axis=1)
x = features
```

```
kaggle_test = pd.DataFrame(test, columns=['body_score', 'inlinks'])
kaggle_test.head()
```

	body_score	inlinks
0	40.023617	4
1	27.538799	1
2	26.976121	2
3	45.848831	5
4	99.811169	11

```
train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.2)
print(train_x.shape, test_x.shape, train_y.shape, test_y.shape)
```

```
(6463, 3) (1616, 3) (6463,) (1616,)
```

2. Build the best classifier you can with the given data, documenting the choices that you make.

a. Try using logistic regression, SVM (multiple kernels), and decision trees.

```
logreg = LogisticRegression().fit(train_x, train_y)
pred1 = logreg.predict(test_x)
cm = confusion_matrix(pred1, test_y)
print(cm)
```

```
[[756 410]
 [110 340]]
```

```
lr = LogisticRegression(solver='lbfgs', max_iter=10000)
lr.fit(train_x, train_y)

#lr_score = lr.score(X_test, y_test)
lr_predictions = lr.predict(test_x)
lr_score = lr.score(test_x, test_y)
cmlr = confusion_matrix(lr_predictions, test_y)
print(cmlr,lr_score)
```

```
[[759 384]
 [119 354]] 0.6887376237623762
```

```
lr_predictions_kaggle = pd.DataFrame(test['row'])
lr_predictions_kaggle['corr'] = lr.predict(kaggle_test)
lr_predictions_kaggle['corr'] = lr_predictions_kaggle['corr'].astype(bool)
lr_predictions_kaggle.to_csv('lr_predictions_kaggle.csv',index=False)
```

```
svm = LinearSVC(max_iter=10000,dual=False)
svm.fit(train_x, train_y)
svm_score = svm.score(test_x, test_y)
svm_predictions = svm.predict(test_x)
cmsvm = confusion_matrix(svm_predictions, test_y)
print('Accuracy of SVM: {:.3f}'.format(svm_score))
```

```
Accuracy of SVM: 0.699
```

```
knn = KNeighborsClassifier(n_neighbors=6)

# Then  fit the model
knn.fit(train_x, train_y)

# How well did we do
knn_score = knn.score(test_x, test_y)
knn_predictions = knn.predict(test_x)
cmknn = confusion_matrix(knn_predictions, test_y)
print('Accuracy of KNN (k = ''): {:.3f}'.format(knn_score))
```

```
Accuracy of KNN (k = ): 0.744
```

```
knn_predictions_kaggle = pd.DataFrame(test['row'])
knn_predictions_kaggle['corr'] = knn.predict(kaggle_test)
knn_predictions_kaggle["corr"] = knn_predictions_kaggle["corr"].astype(bool)
knn_predictions_kaggle.to_csv('knn_predictions_kaggle.csv',index=False)
```

```
kaggle = pd.DataFrame()
print(test_x)
# kaggle['row'] = test_x
```

```
# kaggle['corr'] = knn_predictions
# kaggle.head()
```

```
        body_score  inlinks
6560     50.179601      184
6692    138.343250        4
2796     58.328700       34
4170     26.039544       17
5666     20.669002        1
...            ...      ...
1577     32.819513        1
3823     52.821590       56
2002     41.788673       24
663      72.741179       18
5847     42.338473       33
```

```
dt = DecisionTreeClassifier()

dt.fit(train_x, train_y)

dt_score = dt.score(test_x, test_y)
dt_predictions = dt.predict(test_x)
cmdt = confusion_matrix(dt_predictions, test_y)
print('Accuracy of Decision Tree: {:.3f} '.format(dt_score))
```

```
Accuracy of Decision Tree: 0.690
```

```
dt_predictions_kaggle = pd.DataFrame(test['row'])
dt_predictions_kaggle['corr'] = dt.predict(kaggle_test)
dt_predictions_kaggle["corr"] = dt_predictions_kaggle["corr"].astype(bool)
dt_predictions_kaggle.to_csv('dt_predictions_kaggle.csv',index=False)
```

```
rf = RandomForestClassifier(n_estimators = 22, random_state = 40)

rf.fit(train_x, train_y)

rf_score = rf.score(test_x, test_y)
rf_predictions = rf.predict(test_x)
cmrf = confusion_matrix(rf_predictions, test_y)
print('Accuracy of Random Forest: {:.3f}'.format(rf_score))
```

```
Accuracy of Random Forest: 0.728
```

```
predictions_dictionary = {'Logistic Regression' : lr_predictions, 'KNN' : knn_predictions,
                          'SVM' : svm_predictions, 'Decision Tree' : dt_predictions,
                          'Random Forest' : rf_predictions, 'Actual': test_y}

predictions_df = pd.DataFrame(predictions_dictionary)
predictions_df
```

	Logistic Regression	KNN	SVM	Decision Tree	Random Forest	Actual
6560	0	0	0	0	0	0
6692	1	1	1	1	1	1
2796	0	0	0	0	0	0
4170	0	0	0	1	1	0
5666	0	0	0	0	0	1
...	...	...	...	...	...	...
1577	0	0	0	0	1	0
3823	0	0	0	0	0	0
2002	0	0	0	0	0	0
663	1	0	1	0	0	0
5847	0	0	0	0	0	0

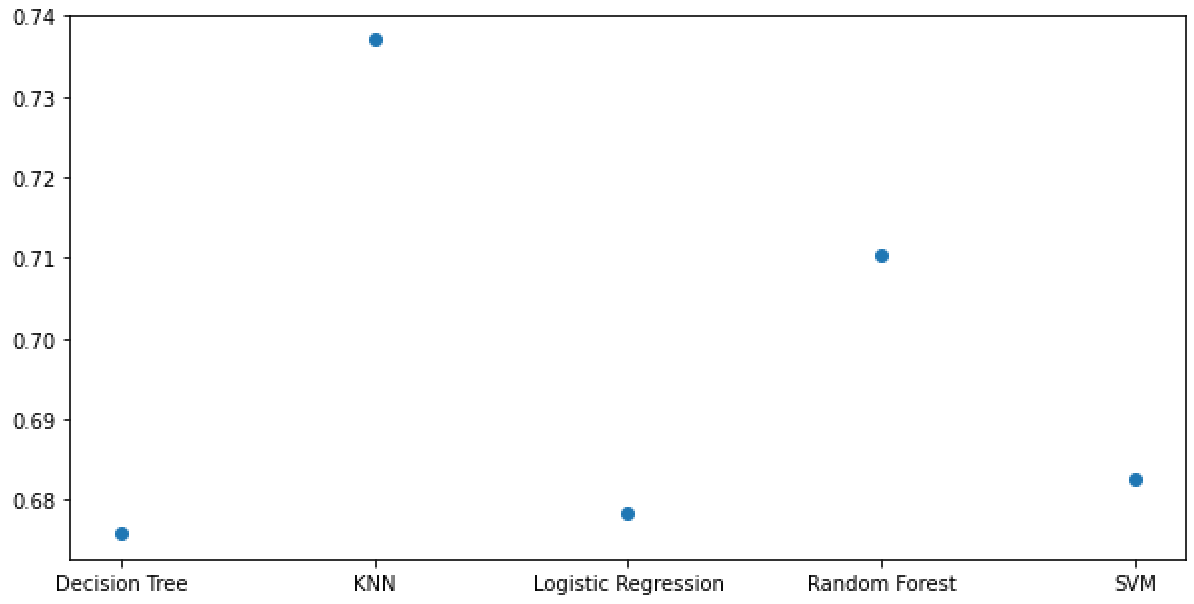
1616 rows × 6 columns

Create a table with your accuracy

```
score_dictionary = {'Logistic Regression' : [lr_score], 'KNN' : [knn_score],
                          'SVM' : [svm_score], 'Decision Tree' : [dt_score],
                          'Random Forest' : [rf_score]}#, 'Actual': test_y}
score_df = pd.DataFrame(score_dictionary)
score_df
```

	Logistic Regression	KNN	SVM	Decision Tree	Random Forest
0	0.678218	0.737005	0.68255	0.675743	0.710396

```
lists = sorted(score_dictionary.items())
scorex, scorey = zip(*lists)
plt.figure(figsize=(10,5))
plt.scatter(scorex, scorey)
plt.show()
```



3. Find additional information you can use to improve predictions. Be creative. Look for features you can extract from the data that you have. NOTE: To get credit for this, you need to have an idea and evaluate it.

Here we choose to add features to train. Some of the features might have categorical variables or might contain NaN or inf values which can be filtered as follows

```
features = pd.DataFrame(train, columns=['body_score', 'inlinks','bsnorm'])
features.replace([np.inf, -np.inf], np.nan, inplace=True)
features.fillna(69, inplace=True)
```

Since variables in collab are global we can change the value of features and run above cells again for updated models.

New feature

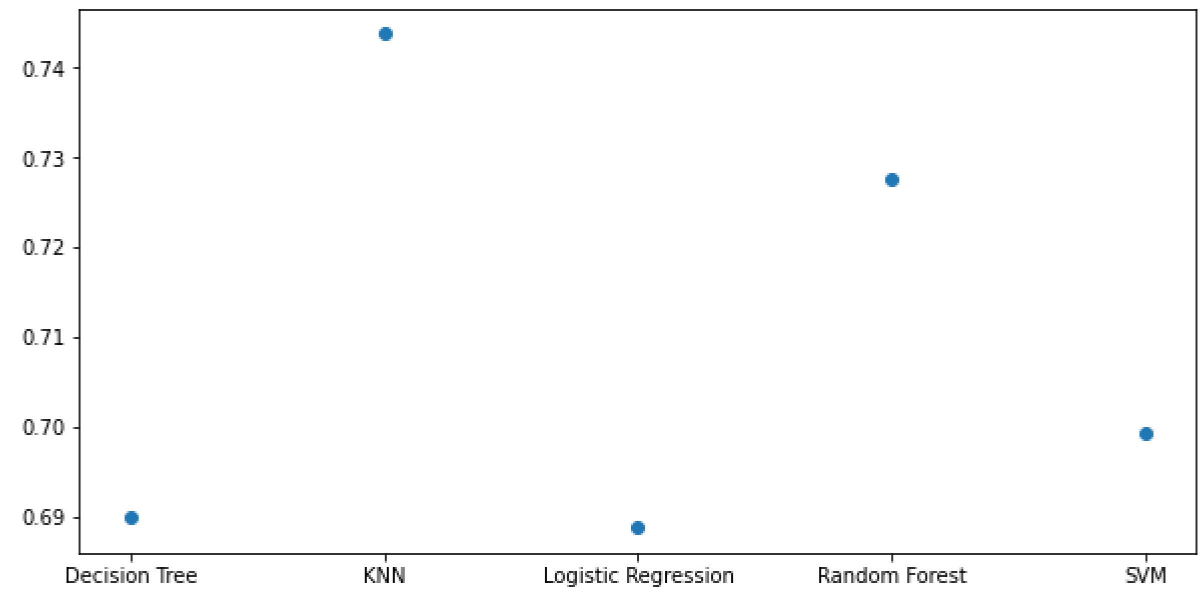
So the feature that is newly made is 'bsnorm' which is normalized body score values which are some of the initial training variables used. The reason these values are normalized is to reduce the outliers and extreme values which result in a better fitting model as seen from the improvement in the accuracies.
So we can surely say that the idea to normalize body score has given us a prositive result which is also present in the test data and can be used to improve the accuracy of all models in this scenario.

Create a table with your accuracy

```
score_dictionary2 = {'Logistic Regression' : [lr_score], 'KNN' : [knn_score],
                          'SVM' : [svm_score], 'Decision Tree' : [dt_score],
                          'Random Forest' : [rf_score]}#, 'Actual': test_y}
score_df2 = pd.DataFrame(score_dictionary2)
score_df2
```

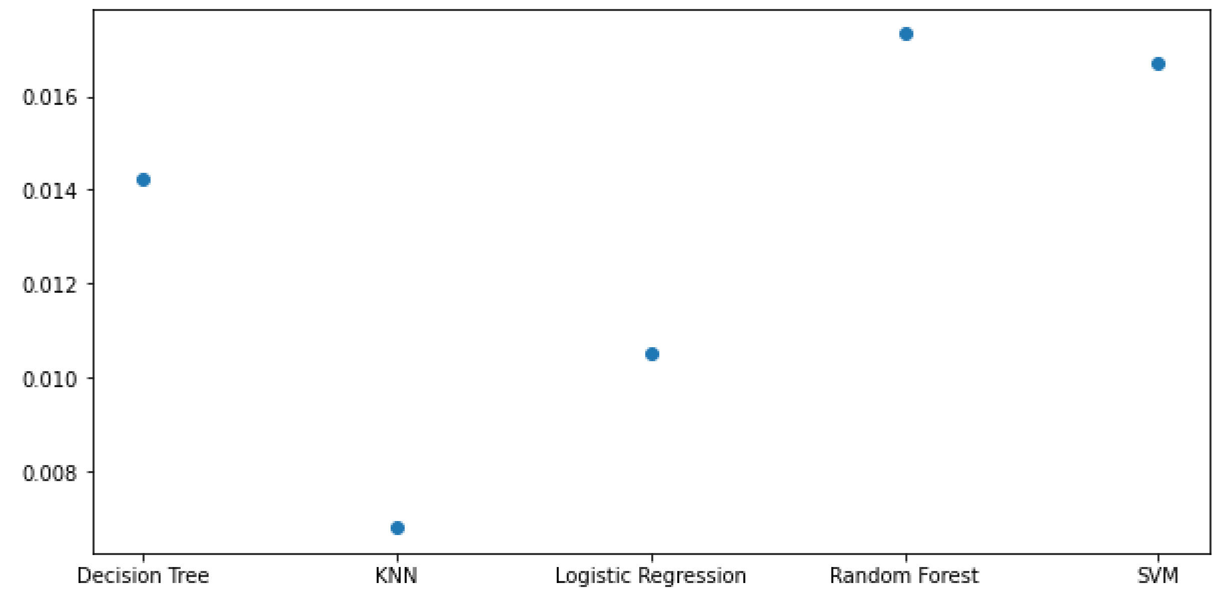
	Logistic Regression	KNN	SVM	Decision Tree	Random Forest
0	0.688738	0.743812	0.699257	0.689975	0.727723

```
lists = sorted(score_dictionary2.items())
scorex, scorey = zip(*lists)
plt.figure(figsize=(10,5))
plt.scatter(scorex, scorey)
plt.show()
```




```
score_differences = {'Logistic Regression' : score_dictionary2['Logistic Regression'][0]-score_dictionary['Logistic Regression'][0]
                    ,'KNN': score_dictionary2['KNN'][0]-score_dictionary['KNN'][0],'SVM': score_dictionary2['SVM'][0]-score_dictionary['SVM'][0],
                    'Decision Tree': score_dictionary2['Decision Tree'][0]-score_dictionary['Decision Tree'][0],
                    'Random Forest': score_dictionary2['Random Forest'][0]-score_dictionary['Random Forest'][0]}
```

```
lists = sorted(score_differences.items())
scorex, scorey = zip(*lists)
plt.figure(figsize=(10,5))
plt.scatter(scorex, scorey)
plt.show()
```



From the above we can clearly see that all values are positive and hence the accuracies have increased by above values from the original accuracies without our added feature and this seems to be significant improvement for adding an additional feature which is derived from existing feature.

4. Challenge: Build a classifier that best predicts correct answers in this dataset. Upload your predictions to Kaggle.

a. Provide your final score and username

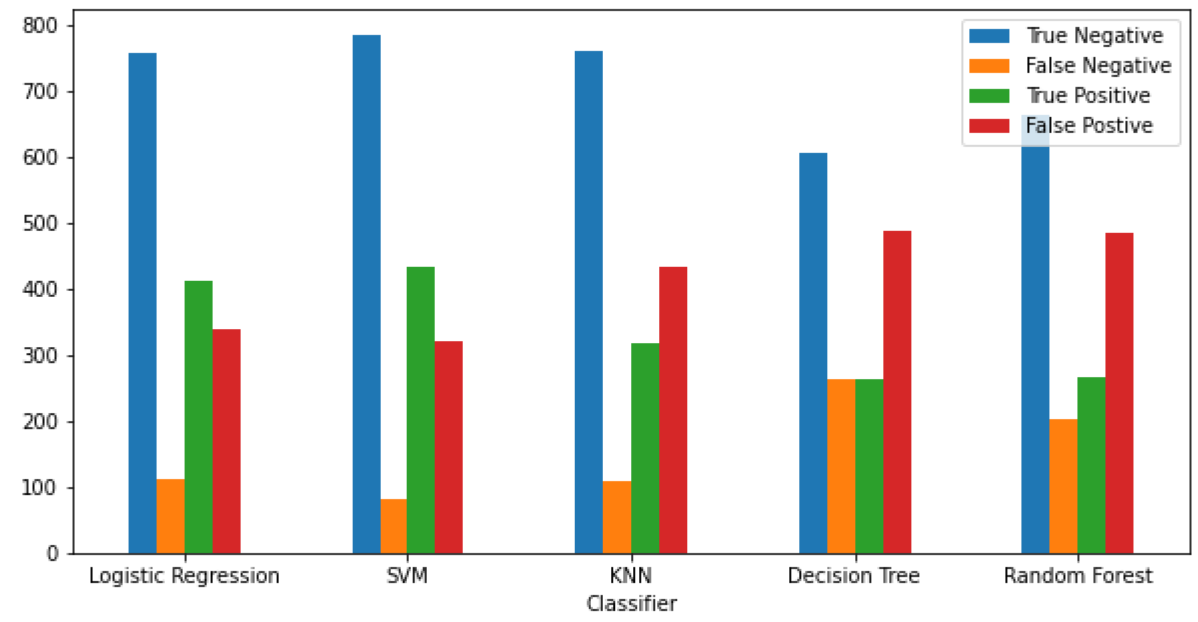
Username : Prasanna Raghavan
Score :0.70806

b. Create an error analysis of your final classifier. Turn this in as “error.pdf”. An error analysis must contain real examples of your data, not just an error matrix

```
confusion = [cmlr, cmsvm, cmknn, cmdt, cmrf]
true_negative = [i[0,0] for i in confusion]
false_negative = [i[1,0] for i in confusion]
true_positive = [i[0,1] for i in confusion]
false_postive = [i[1,1] for i in confusion]

confusion_dict = {'Classifier':['Logistic Regression','SVM','KNN','Decision Tree','Random Forest'],
                  'True Negative':true_negative, 'False Negative':false_negative, 'True Positive':true_positive,'False Postive' : false_postive}
confusion_df = pd.DataFrame(confusion_dict)

confusion_plot = confusion_df.plot(x='Classifier',kind='bar',figsize=(10,5),rot=0)
```



Here we see the error analysis of the classifiers from the confusion matrix of the each classifier and this results in the above bar plot. Here the classifier with a clear distinction between the positives and the negatives is said to be the better classifier as it seems to predict more accurately between the values in comparison to the classifiers with the values that are closer to each other. This latter means that it has trouble differentiating between the values and often gets an inaccurate value. From the above plot Logistic regression, SVM and random forest seem to have clear distinction between their values and give accurate values.