Assignment 6_AN

April 16, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
```

[78]:
```
#pre-processing data tables - data is read into separate files and then␣
↪transposed to have the sample names on the left hand side so that they can␣
↪be fed into the classifier properly

data = pd.read_csv("TCE_feature_table.csv")
labels = pd.read_csv("TCE_metadata.csv")

data_df = pd.DataFrame(data)
data_df = data_df.drop(labels = ["Sample"], axis = 1)
data_df = pd.DataFrame.transpose(data_df)

labels_df = pd.DataFrame(labels)
labels_df = labels_df[["TCE_Exp_Category"]]
labels_df.index = list(data_df.index)

data_df = data_df.fillna('0')

print("Data:\n", data_df)
print("Labels:\n", labels_df)
```

```
Data:
                  0             1              2             3             4  \
X1014   91281.128678  2.036870e+07  100801.729748  28578.671642  64506.008136
X1049  295971.187048  2.364565e+07   147630.84046             0  36993.152512
X1068  244257.923995  2.754199e+07  128838.317479  42871.285598   64365.24175
X1070   82883.827703  2.019781e+07   48201.572085   45854.92046  21970.253718
X1071  357387.911235  2.069844e+07   14503.911432  31862.665469  22431.697984
…                …            …              …             …            …
X2204  149421.602081  2.326146e+07  113216.846541  37778.211692             0
```

X2205	316785.129812	2.252470e+07	175605.835277	15896.028347	41628.632895
X2207	267762.852536	2.074907e+07	58204.870888	0	49299.555895
X2208	432355.324265	2.362341e+07	39426.444151	24436.256522	49749.324282
X2209	295430.297093	1.960256e+07	56656.780775	25098.898492	54592.247426

	5	6	7	8	\
X1014	5.185552e+06	82899.952234	85303.336450	339908.678153	
X1049	6.545665e+06	207861.357053	123532.162386	321187.161276	
X1068	4.849575e+06	50610.656569	91254.969762	322001.526027	
X1070	7.455068e+06	79208.884529	66497.463011	255557.475903	
X1071	4.687812e+06	225161.433069	172721.718845	330914.056524	
…	…	…	…	…	
X2204	5.196808e+06	343604.917933	227029.300686	205799.401347	
X2205	4.112134e+06	217867.885773	165729.808914	270448.56931	
X2207	6.160677e+06	128873.953776	21271.277056	308554.068054	
X2208	3.516818e+06	63992.132779	151167.007784	219020.471954	
X2209	7.362347e+06	154853.48759	136467.473906	218812.683354	

	9	…	7820	7821	7822	\
X1014	118159.943844	…	6447.674642	107738.293977	64090.657502	
X1049	112972.037523	…	0	47826.667193	192932.86854	
X1068	114059.619542	…	0	135320.492745	73408.197627	
X1070	113950.946310	…	77576.329158	204325.331557	92129.587303	
X1071	167858.694497	…	0	254917.232886	271711.245304	
…	…	…	…	…	…	
X2204	102934.555080	…	12640.230049	296358.043729	198007.667272	
X2205	70932.439255	…	8779.899679	157269.406925	209513.842526	
X2207	126307.288853	…	52560.334984	150926.590591	167464.406861	
X2208	94773.004205	…	37417.372146	431501.11746	240766.342223	
X2209	131894.628000	…	9624.708622	270580.515681	151687.440299	

	7823	7824	7825	7826	\
X1014	125098.745799	5.559306e+05	249304.702448	108455.945693	
X1049	54293.215532	8.035466e+05	359059.811526	191538.477009	
X1068	0	2.379774e+05	406754.646264	34895.857662	
X1070	52357.081498	5.069733e+05	294544.798407	151553.922847	
X1071	0	3.953644e+05	321221.583941	0	
…	…	…	…	…	
X2204	0	9.847647e+05	245646.057644	63831.192936	
X2205	0	3.246541e+05	522324.82209	143410.11816	
X2207	44412.144346	8.334720e+05	475176.930347	84766.335535	
X2208	0	1.274749e+06	508574.883772	107625.121444	
X2209	0	1.613409e+06	459051.856371	79754.487721	

	7827	7828	7829
X1014	330854.253142	255148.875874	316429.013152
X1049	552478.386351	513096.539512	196743.192169
X1068	449680.634466	131614.999538	218824.27272

```
X1070  458106.615072  254932.961007   25752.27647
X1071              0              0             0
…                …              …             …
X2204  488267.913873   86101.354318  94260.814032
X2205  502321.406007  486172.178143 183297.295003
X2207  595196.055564  238303.086192  49414.558832
X2208  637043.207002   207608.11336   5939.679989
X2209  502660.817668  260913.200253 214391.820585

[175 rows x 7830 columns]
Labels:
      TCE_Exp_Category
X1014              Low
X1049              Low
X1068              Low
X1070              Low
X1071              Low
…                  …
X2204              Low
X2205              Low
X2207              Low
X2208              Low
X2209              Low

[175 rows x 1 columns]
```

[79]:
```python
#Now we have to convert all the labels from categorical to numerical to assist␣
↪the ML model

labels_df["TCE_Exp_Category"].replace(["Low", "Moderate", "High"], [1, 2, 3],␣
↪inplace = True)
print(labels_df)
```

```
      TCE_Exp_Category
X1014                1
X1049                1
X1068                1
X1070                1
X1071                1
…                  …
X2204                1
X2205                1
X2207                1
X2208                1
X2209                1

[175 rows x 1 columns]
```

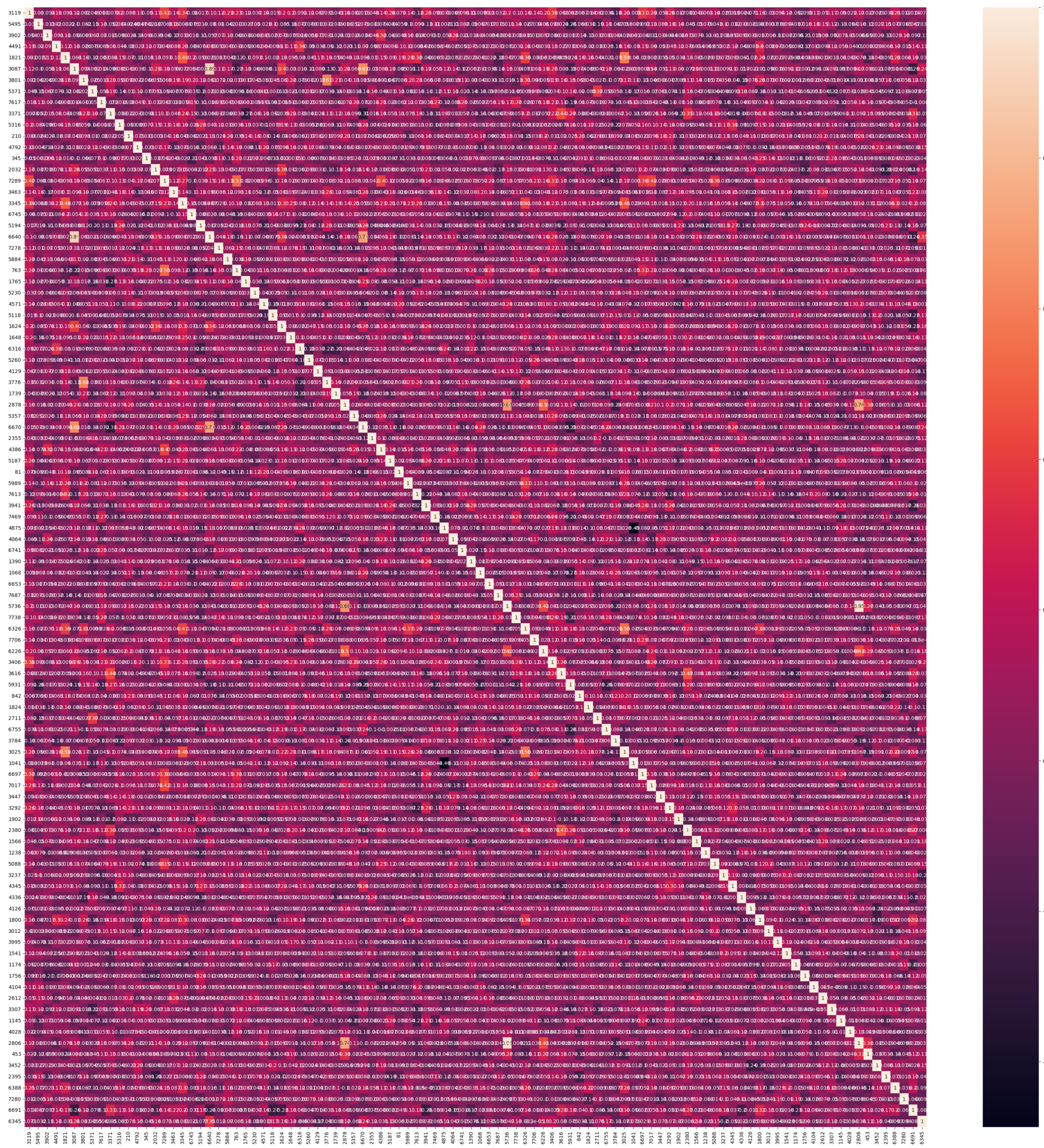
```
[80]: #reducing the number of factors we're looking at by getting rid of any columns␣
  ↪that have a high correlation with another column. This will help us use␣
  ↪columns that have only distinct trends so that it can help with further␣
  ↪classifying our data.

cor_data = data_df.corr()

print(data_df.shape)

for x in cor_data:
    for element in cor_data[x]:
        if element < 1 and element >= 0.80: # we use this as our condition so␣
  ↪we don't get rid of every row. If we set element <= 1, then we would end up␣
  ↪getting rid of every single row as every element correlates perfectly with␣
  ↪itself.
            data_df = data_df.drop(x, axis=1)
            break

print(data_df.shape)

#after this round of processing, we can see that we got rid of approximately␣
  ↪600 rows that had high correlations.
```

```
(175, 7830)
(175, 7264)
```

```
[6]: labels_df['TCE_Exp_Category'].value_counts()
```

```
[6]: 1    95
     3    41
     2    39
     Name: TCE_Exp_Category, dtype: int64
```

```
[65]: #checking correlation to see if we have to perform more feature reduction. It␣
  ↪doesn't look like we have very much overlap between the features so it looks␣
  ↪like we don't have to
plt.figure(figsize=(40,40))
sns.heatmap(data_df.corr(), annot=True)
plt.show()
```



[7]:
```
X_train, X_test, y_train, y_test = train_test_split(data_df, labels_df,␣
    ↪test_size = 0.15, random_state = 20)
print ('The size of our training "X" (input features) is', X_train.shape)
print ('\n')
print ('The size of our testing "X" (input features) is', X_test.shape)
print ('\n')
print ('The size of our training "y" (output feature) is', y_train.shape)
print ('\n')
print ('The size of our testing "y" (output features) is', y_test.shape)
```

```
The size of our training "X" (input features) is (148, 7264)

The size of our testing "X" (input features) is (27, 7264)

The size of our training "y" (output feature) is (148, 1)

The size of our testing "y" (output features) is (27, 1)
```

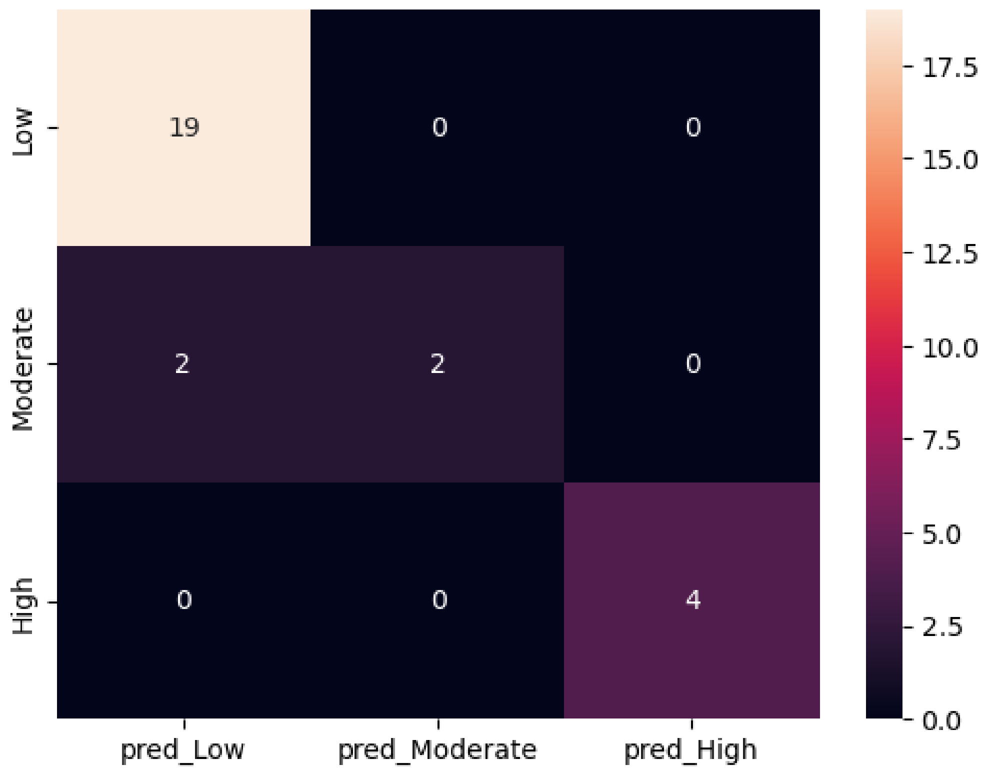
[8]:
```
classifier = RandomForestClassifier(max_features = 0.05, n_estimators = 50,␣
↪max_depth = 200, criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

```
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/3772124414.py:2
: DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
```

[8]:
```
RandomForestClassifier(criterion='entropy', max_depth=200, max_features=0.05,
                       n_estimators=50, random_state=0)
```

[9]:
```
#using the trained model to predict the values given the test data
y_predict = classifier.predict(X_test)
```

[10]:
```
#setting up a visual to understand how many matches and mismatches there are␣
↪between the real data and predicted data
cm = np.array(confusion_matrix(y_test, y_predict, labels=[1,2,3]))
confusion = pd.DataFrame(cm, index=['Low', 'Moderate', 'High'],
                         columns=['pred_Low','pred_Moderate','pred_High'])
sns.heatmap(confusion, annot=True)
plt.show()
```



[10]:
```
print(classification_report(y_test, y_predict))

#these stats are good, a 93% accuracy is the best I've gotten, but there's a␣
↪chance that I'm overfitting the model with the training data available to␣
↪the model. To make it more generalized, I'm going to vary the random state␣
↪of the train and test data split while tuning the parameters to find the␣
↪best model.
```

```
              precision    recall  f1-score   support

           1       0.90      1.00      0.95        19
           2       1.00      0.50      0.67         4
           3       1.00      1.00      1.00         4

    accuracy                           0.93        27
   macro avg       0.97      0.83      0.87        27
weighted avg       0.93      0.93      0.92        27
```

[77]:
```
#this code was developed to run through many random states from 1-20 for the␣
↪training data set. I did this to make sure that no matter how the data is␣
↪split, the model still maintains a high accuracy for the prediction of data
```

```
from statistics import mean
accuracy_list = []

for x in range(20):
    X_train, X_test, y_train, y_test = train_test_split(data_df, labels_df,␣
↪test_size = 0.20, random_state = x)
    classifier = RandomForestClassifier(max_features = 0.3, n_estimators = 100,␣
↪max_depth = 20, min_samples_leaf = 8, criterion = 'entropy', random_state =␣
↪0)
    classifier.fit(X_train, y_train)
    y_predict = classifier.predict(X_test)
    rep = classification_report(y_test,y_predict).split("\n")
    for num in rep[6].split(" "):
        if "." in num:
            accuracy_list.append(float(num))

print(mean(accuracy_list))
print(accuracy_list)
```

```
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
```

```
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
```

```
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
/var/folders/z2/z4rmk79963l_trmcdlb01zsw0000gn/T/ipykernel_26109/446282627.py:8:
DataConversionWarning: A column-vector y was passed when a 1d array was
expected. Please change the shape of y to (n_samples,), for example using
ravel().
  classifier.fit(X_train, y_train)
```

```
0.781
[0.77, 0.89, 0.71, 0.83, 0.86, 0.86, 0.71, 0.77, 0.66, 0.8, 0.77, 0.8, 0.86,
0.66, 0.71, 0.71, 0.83, 0.8, 0.71, 0.91]
```

[ ]:
```
# Here are some of the accuracies I saw while completing parameter tuning. With
 ↪the final set of parameters, I found
# that they would yield the highest accuracies regardless of the random states
 ↪that were applied to the train/test data
# attribution. There are many reasons why the accuracy is not consistently at a
 ↪high level. One possiblity could be
# that the data has a lot of NaN values which was replaced by a 0 in
 ↪pre-processing. These 0's could be messing with
# the model development and causing lower accuracy levels than normal. Another
 ↪reason could be that not enough
# or too much feature reduction was done which caused a high or low accuracy
 ↪than in reality. Overall, however,
# accuracies ranging in the high 70s, low 80s percentage-wise is relatively
 ↪good as a tool that can assist scientists in
# making more accurate decisions on what a patient's TCE exposure could be.

# max_features = 0.5, n_estimators = 50, max_depth = 100 acc = 0.76
# max_features = 0.5, n_estimators = 100, max_depth = 200 acc = 0.785
```

```
# max_features = 0.3, n_estimators = 100, max_depth = 20 acc = 0.788
# max_features = 0.3, n_estimators = 100, max_depth = 20, min_samples_leaf = 8 acc = 0.782
```