GROUP 82

Esha Kolte(180020031) & Jyotirmoy Roy(180020044)

Introduction

The peptides that distinguish grade I meningioma from higher grades are still not properly discovered. In other diseases , normal statistical methods like fold change criteria and p values help in finding the differentially expressed proteins for that disease. However in meningioma samples, the variations in the samples are so much that normal statistical tests do not give reliable results. So we used unsupervised clustering in the form of PCA to find the topmost features and then used these topmost peptides in designing the ML model to separate a severe patient from a mild patient.

Libraries/Frameworks Used:

Pandas, Numpy, matplotlib, seaborn, scikit-learn

Implementation Details

For SM_MV dataset:

Loading of the Dataset

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from matplotlib.cm import register_cmap
from scipy import stats
from sklearn.decomposition import PCA
import seaborn
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
path = "drive/My Drive/SM_MV_.csv"
df1 = pd.read_csv(path)
```

```
df1.head(3)
```

	Sample	LFQ_Int_CP35491	LFQ_Int_CP10882	LFQ_Int_CN06567	LFQ_Int_CP16894	LFQ_Int_CP22324	LFQ_Int_CH10571	LFQ_Int_CN30310
0	Label	High Grade	High Grade	High Grade	High Grade	High Grade	High Grade	High Grade
1	O00170	1002400	264920000	77973000	53329000	35962000	29680000	27150000
2	O00203	19582000	277510000	70955000	44511000	46226000	90544000	49622000

Scaling of Dataset

```
[29] df_robust = pd.DataFrame(StandardScaler().fit_transform(df1), columns=df1.columns)
     df_robust.head(3)
```

Sample	000170	000203	000233	000264	000299	014773	014818	015143	015144	015145	015173	015230	043143
0	-0.897577	-0.638089	-0.918204	-0.759059	-0.993980	-0.936530	-0.557747	-0.920684	-1.180337	-0.833066	-0.770158	-0.619115	-0.716727
1	3.611132	2.980736	-0.698088	0.101923	1.098640	2.444960	1.515920	2.173013	2.018392	2.415767	1.681002	3.690813	2.838743
2	0.417371	0.082693	0.119870	0.715721	1.200521	1.223655	1.067382	0.121267	1.167368	0.434553	0.228584	-0.245229	0.103807

3 rows × 2797 columns

Principal Component Analysis

Covariance matrix

```
[[ 1.05        0.90736153  0.2911117  ...  0.59821483  0.01100223
  -0.07060035]
 [ 0.90736153  1.05        0.47043016 ...  0.62413063 -0.03268425
  -0.09215782]
 [ 0.2911117   0.47043016  1.05       ...  0.5934488   0.05320153
  -0.16789642]
 ...
 [ 0.59821483  0.62413063  0.5934488  ...  1.05       -0.17401709
  -0.27470341]
 [ 0.01100223 -0.03268425  0.05320153 ... -0.17401709  1.05
   0.94057355]
 [-0.07060035 -0.09215782 -0.16789642 ... -0.27470341  0.94057355
   1.05      ]]
```

Eigenvectors

```
Eigenvectors
[[ 2.36817545e-02+0.00000000e+00j  1.59173650e-02+0.00000000e+00j
  -1.14564273e-02+0.00000000e+00j ...  4.50371482e-03-3.46751650e-03j
   5.47342212e-03-1.11075501e-05j  5.47342212e-03+1.11075501e-05j]
 [ 2.49499986e-02+0.00000000e+00j  3.63859228e-03+0.00000000e+00j
  -1.79440398e-03+0.00000000e+00j ...  1.97050816e-04-3.02417497e-05j
   1.55178288e-04-7.56603169e-05j  1.55178288e-04+7.56603169e-05j]
 [ 1.57833868e-02+0.00000000e+00j -3.40909363e-02+0.00000000e+00j
   1.72288496e-02+0.00000000e+00j ... -8.12826805e-05-2.76845899e-04j
  -3.90688888e-04-1.47802460e-04j -3.90688888e-04+1.47802460e-04j]
 ...
 [ 1.82648009e-02+0.00000000e+00j  1.05128091e-03+0.00000000e+00j
  -2.11753707e-02+0.00000000e+00j ...  3.20769323e-03+2.32214580e-02j
   1.52920975e-02-4.05207716e-03j  1.52920975e-02+4.05207716e-03j]
 [ 2.05903348e-03+0.00000000e+00j  2.91035906e-02+0.00000000e+00j
   5.58163862e-02+0.00000000e+00j ... -9.88852053e-03+1.06821345e-02j
   1.55627716e-03-4.28813673e-03j  1.55627716e-03+4.28813673e-03j]
 [-1.72485960e-03+0.00000000e+00j  4.31685081e-02+0.00000000e+00j
   4.88025946e-02+0.00000000e+00j ...  2.42930450e-02+2.51401986e-02j
   4.76628119e-02-1.97499234e-02j  4.76628119e-02+1.97499234e-02j]]
```

Eigenvalues

	Eigenvalues
0	1.415313e+03+0.000000e+00j
1	2.397867e+02+0.000000e+00j
2	2.159604e+02+0.000000e+00j
3	1.810217e+02+0.000000e+00j
4	1.321263e+02+0.000000e+00j
...	...
2792	6.799076e-17+0.000000e+00j
2793	2.444043e-16+9.906640e-17j
2794	2.444043e-16-9.906640e-17j
2795	4.656300e-18+2.053894e-16j
2796	4.656300e-18-2.053894e-16j

2797 rows × 1 columns

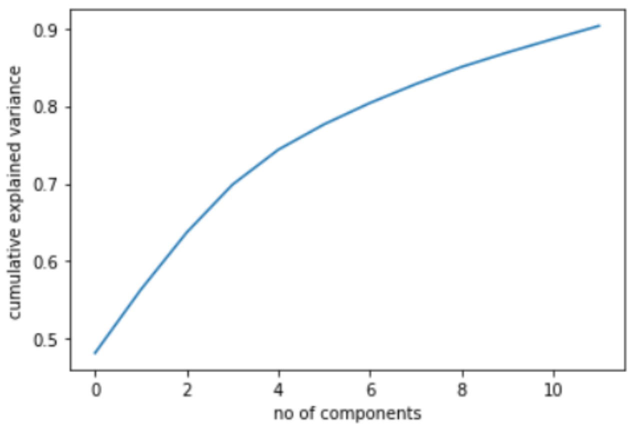
Principal Component Analysis

```
pca = PCA(n_components=2)
pca.fit_transform(df_robust)
```

```
array([[-2.90640108e+01, -1.85214723e+00],
       [ 9.39183908e+01,  4.37723650e+01],
       [ 1.82420067e+01, -1.04072630e+01],
       [ 7.58310742e+00, -4.24222764e+00],
       [-9.57029076e+00, -2.33869242e+00],
       [-5.79625688e+00, -2.64628453e+00],
       [-1.68723236e+01, -3.04415479e+00],
       [-2.20448330e+01, -5.51630617e-01],
       [-2.74253371e+01,  4.01174959e+01],
       [-2.06657324e+01, -9.81147838e-01],
       [ 1.83237319e+01, -8.89992590e+00],
       [-3.73777171e+01,  1.99914913e+00],
       [-3.99414887e+01,  5.58365210e-01],
       [-2.77796041e+01,  3.64157526e-02],
       [-3.31136443e+01, -1.64713903e-02],
       [ 8.68338280e+01, -2.90476898e+01],
       [-2.02410130e+01,  5.65226193e-01],
       [ 5.40452309e+01, -8.69603352e-01],
       [ 5.46148735e+00, -7.98761778e+00],
       [-2.89605379e+00, -3.50585711e+00],
       [ 8.38052227e+00, -1.06583039e+01]])
```

Plot of cumulative explained variance vs number of components

```
pca = PCA(n_components=0.9).fit(X_std)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('no of components')
plt.ylabel('cumulative explained variance')
plt.show()
```




```
pca.n_components_
```

```
12
```

```
pca.explained_variance_ratio_
```

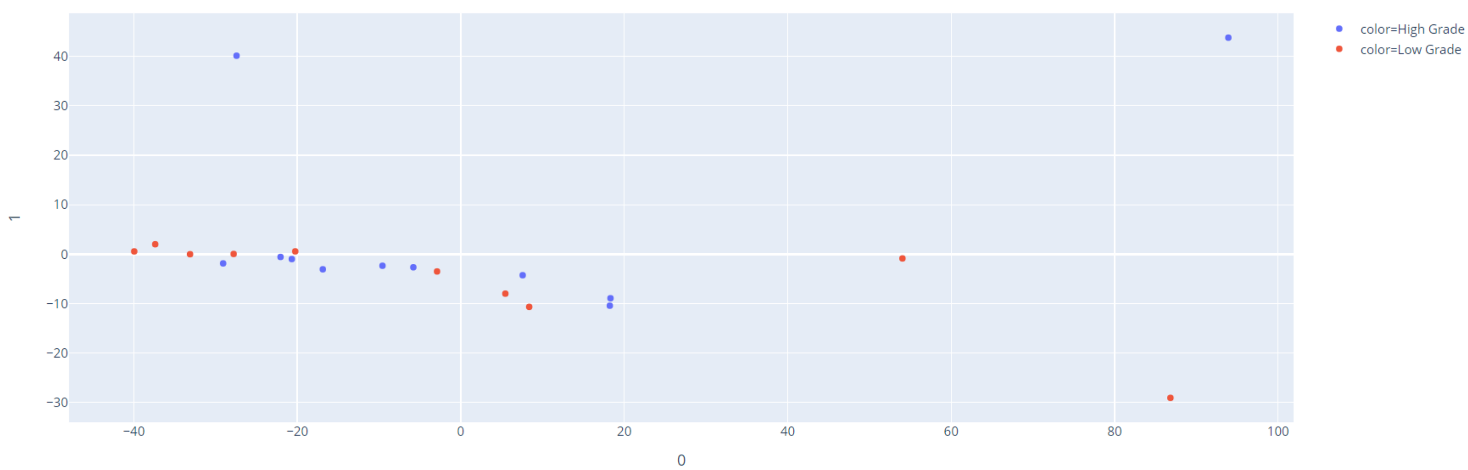
```
array([0.48191523, 0.08164759, 0.0735347 , 0.06163805, 0.04498911,
       0.03268346, 0.02747106, 0.02430601, 0.022107  , 0.01862011,
       0.01734761, 0.01685294])
```

Features in the order of their importance

	Sum
Sample	
O43175	0.330887
P09471	0.329201
Q5THK1	0.327736
Q86UU1	0.326203
P60201	0.315361
...	...
O14683	0.237077
Q13976	0.236923
P00966	0.236838
O75122	0.236779
Q14161	0.236706

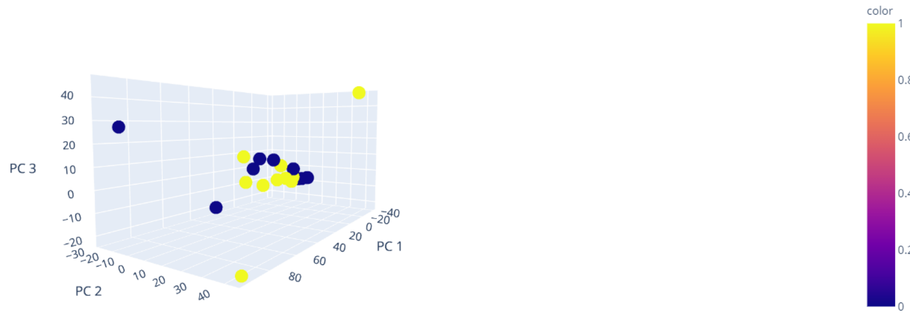
300 rows × 1 columns

2D PCA Scatter Plot



3D PCA Scatter Plot

Total Explained Variance: 63.71%



For DB_MV dataset:

Loading of Dataset

```
[ ] import pandas as pd
    import numpy as np
    from sklearn.preprocessing import StandardScaler
    import matplotlib.pyplot as plt
    from matplotlib.cm import register_cmap
    from scipy import stats
    from sklearn.decomposition import PCA
    import seaborn
```

```
[ ] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[ ] path = "drive/My Drive/DB_MV.csv"
    df1 = pd.read_csv(path)
```

```
[ ] df1.head(3)
```

	Sample	22324	34759	7938	41	9048	50458	15649	31148	34915	14619	22466	30755	3080
0	Class	High Grade	High Grade	High Grade	High Grade	High Grade	High Grade	High Grade	High Grade	High Grade	High Grade	Low Grade	Low Grade	Low Grade
1	E9PAV3	33186000	6409600	21822000	55404000	19362000	13532000	12474000	31747000	4545100	61812000	16580000	42177000	63247000
2	O00170	19059000	1870700	6457100	31532000	8949600	9782500	2633800	12036000	1936800	14601000	3532700	10497000	15675000

Scaling of Dataset

```
[ ] df_robust = pd.DataFrame(scaler.fit_transform(df1), columns=df1.columns)
    df_robust.head(3)
```

Sample	E9PAV3	000170	000231	000232	000264	000299	000410	000429	000571	000764	014579	014773
0	0.423966	1.270458	1.671208	1.288838	0.386745	1.441948	10.597874	2.540299	1.717557	1.017513	1.553079	1.581909
1	-0.803971	-1.083556	0.302380	2.170666	-0.614010	0.585462	-0.333148	-0.094388	-0.366166	-0.319752	-0.145535	-0.100821
2	-0.097175	-0.455428	-0.022844	0.387268	-0.828163	-0.320303	-0.307432	0.801937	-0.004925	0.000000	0.000000	0.029497

3 rows × 2409 columns

Principal Component Analysis
Covariance matrix

```
[[1.04166667 0.80544836 0.60631416 ... 0.47576788 0.6711079  0.57232503]
 [0.80544836 1.04166667 0.54434854 ... 0.16517416 0.67518013 0.29013842]
 [0.60631416 0.54434854 1.04166667 ... 0.23606174 0.8207803  0.6519101 ]
 ...
 [0.47576788 0.16517416 0.23606174 ... 1.04166667 0.46464102 0.14195845]
 [0.6711079  0.67518013 0.8207803  ... 0.46464102 1.04166667 0.42230647]
 [0.57232503 0.29013842 0.6519101  ... 0.14195845 0.42230647 1.04166667]]
```

Eigenvectors

```
Eigenvectors
[[ 2.36911945e-02+0.j         -5.14638372e-03+0.j
   3.14410817e-02+0.j         ... -1.43144597e-02+0.00159898j
  -1.43144597e-02-0.00159898j  1.31431056e-02+0.j        ]
 [ 2.16919239e-02+0.j          1.26867676e-02+0.j
   1.40846847e-02+0.j         ... -5.66399342e-04-0.00026668j
  -5.66399342e-04+0.00026668j  3.74968565e-04+0.j        ]
 [ 2.70407587e-02+0.j          1.68382977e-02+0.j
   5.82707306e-03+0.j         ...  3.45746443e-04-0.00062978j
   3.45746443e-04+0.00062978j -6.69889938e-04+0.j        ]
 ...
 [ 1.73993030e-02+0.j         -4.40323446e-02+0.j
  -1.34476023e-03+0.j         ...  8.18504560e-05+0.00415748j
   8.18504560e-05-0.00415748j  2.14053820e-02+0.j        ]
 [ 2.89088532e-02+0.j          3.03204255e-03+0.j
  -8.64988533e-03+0.j         ...  1.95802319e-02+0.00283252j
   1.95802319e-02-0.00283252j -2.73359155e-02+0.j        ]
 [ 1.64707707e-02+0.j          7.27793538e-03+0.j
   2.54563282e-02+0.j         ...  5.84377718e-03+0.00218779j
   5.84377718e-03-0.00218779j  2.53742504e-03+0.j        ]]
```

Eigenvalues

	Peptides	Eigen_values
0	E9PAV3	1.046284e+03
1	O00170	3.414690e+02
2	O00231	1.535777e+02
3	O00232	1.389698e+02
4	O00264	1.311891e+02
...	...	...
2404	Q9UL45	1.663060e-16
2405	Q9UNF1	1.663060e-16
2406	Q9UPA5	7.884231e-17
2407	Q9Y512	7.884231e-17
2408	Q9Y657	1.861487e-16

2409 rows × 2 columns

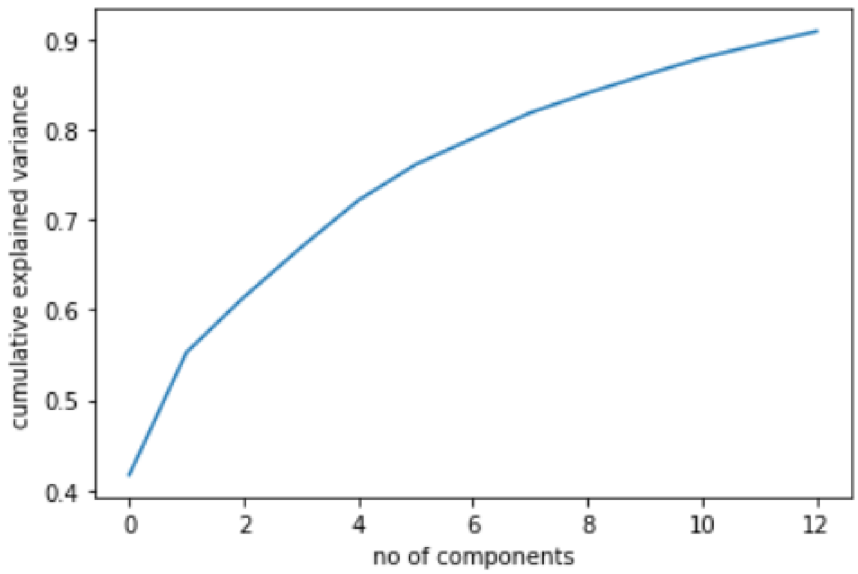
Principal Component Analysis

```
pca = PCA(n_components=2)
pca.fit_transform(df_robust)
```

```
array([[-4.35292582e+00, -8.32087986e+01],
       [-9.87338124e+01, -2.63247033e+01],
       [-2.95021874e+01,  2.93936901e+02],
       [-1.34764314e+02, -1.73162827e+01],
       [-1.41751076e+02, -9.57427669e+00],
       [-1.41361666e+02, -6.56291916e+00],
       [-1.38486819e+02,  8.48283752e+00],
       [-1.41937316e+02, -6.03470678e+00],
       [-1.44860476e+02, -9.74378774e+00],
       [ 3.02996470e+03, -6.05360429e+00],
       [-1.39704098e+02, -8.56891511e+00],
       [-1.38045166e+02, -1.46390953e+01],
       [-1.38854407e+02, -4.06979716e+00],
       [-1.18036074e+02, -1.79300401e+01],
       [-1.46253148e+02, -5.20707957e+00],
       [-1.36510820e+02, -2.07434301e+00],
       [-1.41276548e+02, -1.20124507e+01],
       [-1.40265336e+02, -1.02425150e+01],
       [-1.07177625e+02, -1.85550207e+01],
       [-1.42435696e+02, -8.37660741e+00],
       [-1.43240326e+02, -8.40760854e+00],
       [-1.40226403e+02, -9.53447716e+00],
       [-1.40712886e+02, -1.11047331e+01],
       [-1.45829436e+02, -4.34534587e+00],
       [-1.35646142e+02, -2.53263027e+00]])
```

Plot of cumulative explained variance vs number of components

```
pca = PCA(n_components=0.9).fit(X_std)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('no of components')
plt.ylabel('cumulative explained variance')
plt.show()
```




```
pca.n_components_
```

```
13
```

```
pca.explained_variance_ratio_
```

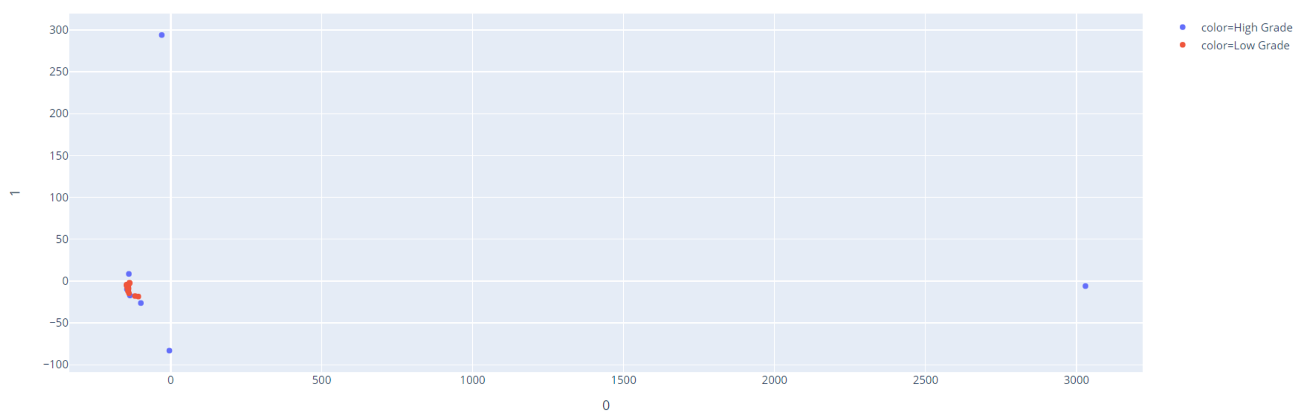
```
array([0.41695014, 0.13607729, 0.06120158, 0.05538026, 0.0522796 ,
       0.03976235, 0.02904001, 0.02847495, 0.02181155, 0.01997979,
       0.01904819, 0.01510468, 0.01455655])
```

Features in the relative order of their importance

Sample	Sum
P00915	0.370568
P11277	0.369063
P68871	0.360195
P35523	0.356413
P02042	0.355986
...	...
Q9BTT0	0.268464
O15127	0.268406
Q13542	0.268306
P42766	0.268192
O14558	0.268128

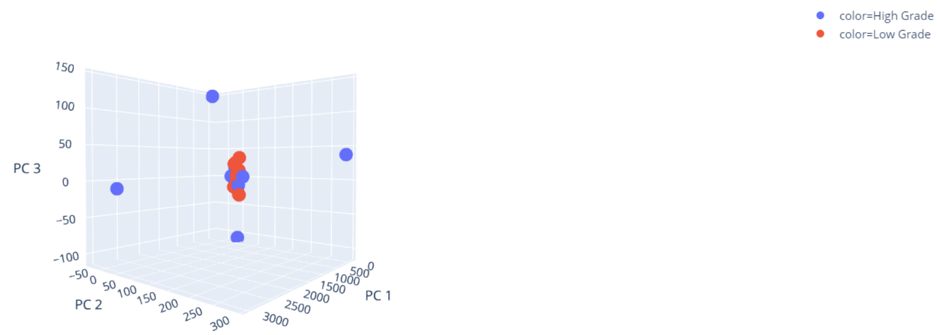
300 rows × 1 columns

2D PCA Scatter Plot



3D PCA Scatter Plot

Total Explained Variance: 98.24%



Top 40 important features on which Random Forest algorithm was applied

```
Index(['Unnamed: 0', 'A0A0B4J1X5', 'O00592', 'O15438', 'O75884', 'O95816',
       'P00747', 'P00966', 'P01861', 'P02652', 'P02671', 'P02675', 'P02679',
       'P02730', 'P02750', 'P02753', 'P02766', 'P05090', 'P08311', 'P08697',
       'P12429', 'P14923', 'P17677', 'P19652', 'P19827', 'P22105', 'P25311',
       'P25685', 'P26447', 'P35556', 'P39060', 'P54652', 'P61626', 'Q13510',
       'Q14126', 'Q15714', 'Q96CN7', 'Q96T23', 'Q9BWS9', 'Q9UM54', 'Q9Y625'],
```

Random Forest Algorithm on Combined Dataset

Loading of Dataset

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
path = "drive/My Drive/CombinedInput.xlsx"
df = pd.read_excel(path)
```

```
df.head(3)
```

	Unnamed: 0	A0A0B4J1X5	O00592	O15438	O75884	O95816	P00747	P00966	P01861	P02652	P02671	P02675	P02679	P02730
0	1	0.482161	0.473002	0.509935	0.545214	0.535377	0.507855	0.572658	0.627768	0.572704	0.622162	0.646808	0.630159	0.551593
1	1	0.611093	0.495021	0.611706	0.464316	0.585244	0.585918	0.557084	0.703404	0.624437	0.724484	0.753363	0.746158	0.619233
2	1	0.544422	0.508639	0.699819	0.467568	0.505753	0.667078	0.624651	0.662358	0.723317	0.772431	0.778523	0.775723	0.738656

Random Forest Classifier

```
classifier = RandomForestClassifier(n_estimators=10, max_depth=5, max_features='sqrt', random_state = 42)
classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=5, max_features='sqrt',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=42, verbose=0,
                       warm_start=False)
```

Classification Report with accuracy

```
Classification Report:
              precision    recall  f1-score   support

           0       0.58      0.88      0.70         8
           1       0.50      0.17      0.25         6

    accuracy                           0.57        14
   macro avg       0.54      0.52      0.48        14
weighted avg       0.55      0.57      0.51        14

Accuracy: 0.5714285714285714
```

Metrics for Random Forest model evaluation

```
from sklearn.metrics import accuracy_score
scores_classification = accuracy_score(y_test, y_pred)
print(scores_classification)
```

```
0.5714285714285714
```

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[7 1]
 [5 1]]
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt
train_preds = classifier.predict(X_train)
mse = mean_squared_error(y_train, train_preds)
rmse = sqrt(mse)
rmse
```

```
0.1767766952966369
```

```
test_preds = classifier.predict(X_test)
mse = mean_squared_error(y_test, test_preds)
rmse = sqrt(mse)
rmse
```

```
0.6546536707079771
```

KNN Algorithm on Combined Dataset

Model and its metrics

```
# Creating the Training and Test set from data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.30, random_state = 42)

# Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=3)
```

```
print(classifier.score(X_test, y_test))
```

```
0.5
```

```
scores_classification = accuracy_score(y_test, y_pred)
print(scores_classification)
```

```
0.5
```

```
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[6 2]
 [5 1]]
```

```
train_preds = classifier.predict(X_train)
mse = mean_squared_error(y_train, train_preds)
rmse = sqrt(mse)
rmse
```

```
0.3535533905932738
```

```
test_preds = classifier.predict(X_test)
mse = mean_squared_error(y_test, test_preds)
rmse = sqrt(mse)
rmse
```

```
0.7071067811865476
```

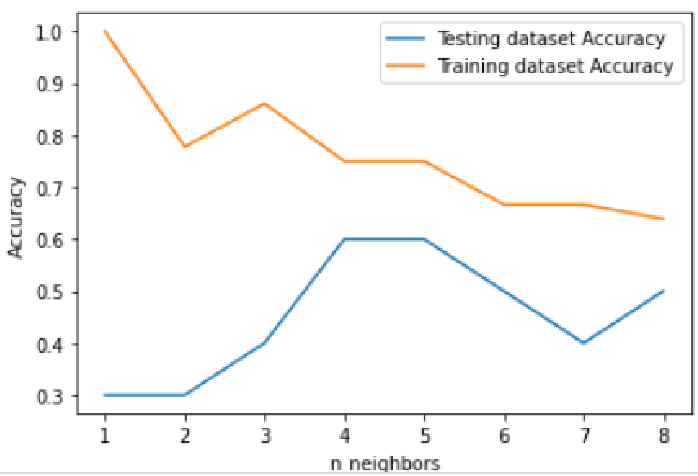
```python
X_train, X_test, y_train, y_test = train_test_split(
             X, y, test_size = 0.2, random_state=42)

neighbors = np.arange(1, 9)
train_accuracy = np.empty(len(neighbors))
test_accuracy = np.empty(len(neighbors))

# Loop over K values
for i, k in enumerate(neighbors):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)

    # Compute traning and test data accuracy
    train_accuracy[i] = knn.score(X_train, y_train)
    test_accuracy[i] = knn.score(X_test, y_test)

# Generate plot
plt.plot(neighbors, test_accuracy, label = 'Testing dataset Accuracy')
plt.plot(neighbors, train_accuracy, label = 'Training dataset Accuracy')
plt.legend()
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy')
plt.show()
```



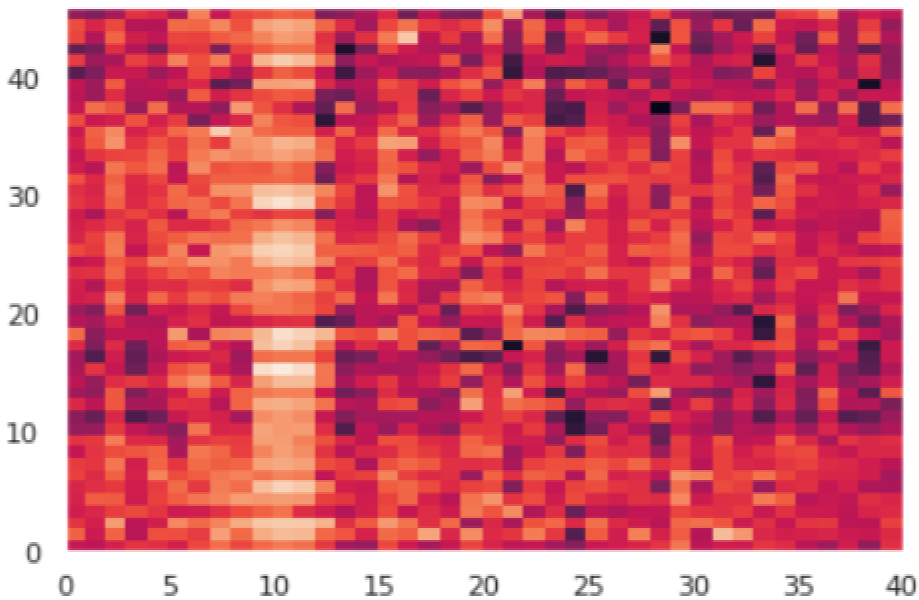

```python
print(pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predicted']))
```

```
Predicted  0  1
Actual
0          5  1
1          4  0
```

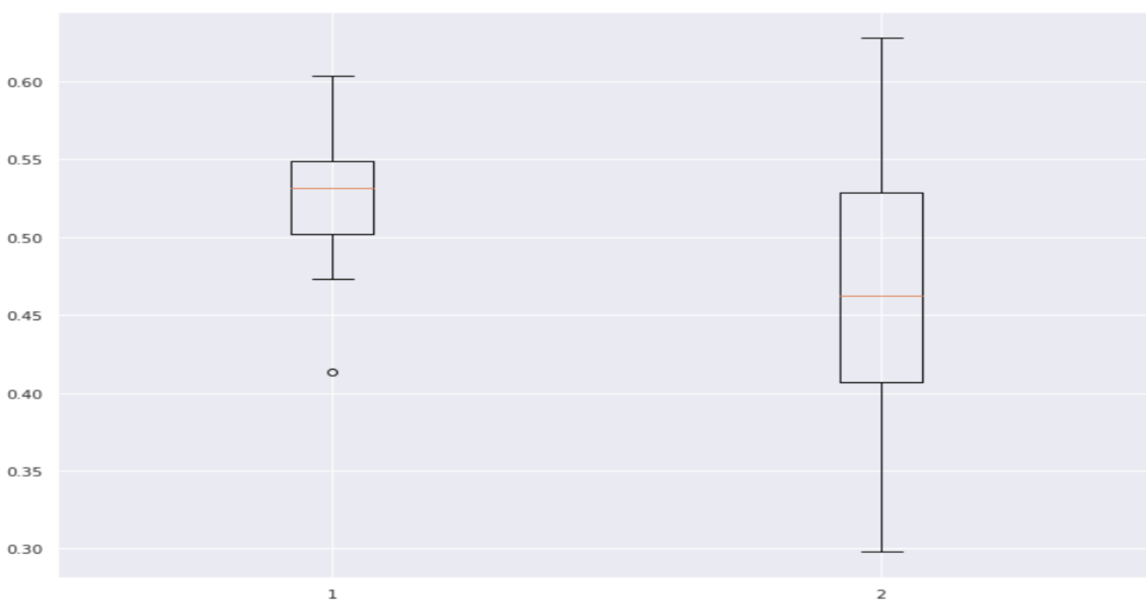
```python
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.5
```

Heatmap



Boxplot for High Grade Vs Low Grade for Peptide O00529:



1:High Grade 2:Low Grade

Results:

- 40 important peptides were identified which were common across the datasets using the concept of PCA and eigenvectors
- These features were used to train Random forest and KNN models to classify the grade of brain tumor correctly.
- With the Random Forest algorithm, an accuracy of 57.14% was achieved and with KNN, an accuracy of 50% was achieved.
- Although the results aren't satisfactory , this was a marked improvement from the traditional statistical analysis(using p value and fold change)
- The number of important common peptides identified **increased by 4 folds** and accuracy of Machine learning models using these features **increased from 35% to 57.14%**