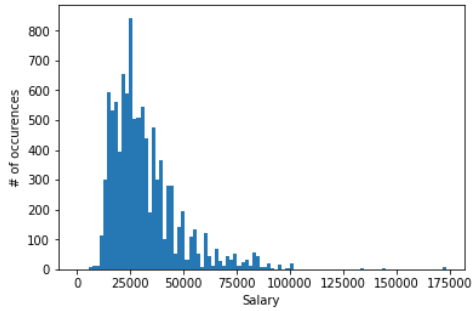
STAT 517 PROJECT 2

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

Question 0: Salary Prediction

Preliminary Visualization

```
salary = originalsalary
plt.hist(originalsalary['SalaryNormalized'], bins=[n*1750 for n in range(1
00)], histtype='stepfilled')
plt.xlabel("Salary")
plt.ylabel("# of occurences")
plt.show()
```



Data Preprocessing

```
# preprocessing

#get rid of useless columns
salary = salary.drop(['LocationRaw', 'SalaryRaw', 'SourceName'], axis=1)

#this loop does preprocessing on columns that are strings
for n in salary.columns:
    if salary[n].dtype == 'object':
        salary[n] = salary[n].str.lower()
        salary[n] = salary[n].str.strip()

#replace null values with the mode (most frequent occurence in the feature)
salary['Title'].fillna(value=salary['Title'].value_counts().axes[0][0], inplace=True)
salary['ContractType'].fillna(value=salary['ContractType'].value_counts().axes[0][0], inplace=True
)
salary['ContractTime'].fillna(value=salary['ContractTime'].value_counts().axes[0][0], inplace=True
)
salary['Company'].fillna(value=salary['Company'].value_counts().axes[0][0], inplace=True)
print (salary.shape)
```

```
(10000, 9)
```

```
#tf-idf
#first we create vector using only 1000 words and removing useless stop words
#we just consider words that matter to the target variable (salary)
vector = TfidfVectorizer(stop_words='english', max_features=1000)
titleVectorized = pd.DataFrame(vector.fit_transform(salary.Title).toarray(), columns = vector.get_
feature_names())
descriptionVectorized = pd.DataFrame(vector.fit_transform(salary.FullDescription).toarray(), colum
ns = vector.get_feature_names())
companyVectorized = pd.DataFrame(vector.fit_transform(salary.Company).toarray(), columns = vector.
get_feature_names())
categoryVectorized = pd.DataFrame(vector.fit_transform(salary.Category).toarray(), columns = vecto
r.get_feature_names())
#salary = salary.drop('FullDescription', axis=1)
#salary = pd.concat([salary.drop(['Title','FullDescription','Company','Category'], axis=1), titleV
ectorized,descriptionVectorized,companyVectorized,categoryVectorized], axis=1)
print (salary.shape)
print (salary.drop(['Title','FullDescription','Company','Category'], axis=1).shape)
print (titleVectorized.shape)
print (descriptionVectorized.shape)
print (companyVectorized.shape)
print (categoryVectorized.shape)
```

```
(10000, 9)
(10000, 5)
(10000, 1000)
(10000, 1000)
(10000, 1000)
(10000, 45)
```

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

```
#preprocess categorical features
#salary = pd.get_dummies(salary, columns=['Title', 'LocationNormalized', 'ContractTime', 'Contract
Type', 'Company', 'Category'], prefix=['Title', 'LocationNormalized', 'ContractTime', 'ContractTyp
e', 'Company', 'Category'])
salary = pd.get_dummies(salary, columns=['LocationNormalized', 'ContractTime', 'ContractType'], pr
efix=['LocationNormalized', 'ContractTime', 'ContractType'])
print (salary.shape)
```

Training and Testing Models

```
#split dataset into training and testing 80% 20%
X = salary.drop('SalaryNormalized', axis=1)
Y = salary.SalaryNormalized
pca = PCA(n_components=1000)
pca.fit(X)
X = pca.transform(X)
xTrain, xTest, yTrain, yTest = train_test_split(X, Y, random_state=int(time.time()), test_size=0.2
0)
```

Linear Regression

```
lr = LinearRegression().fit(xTrain, yTrain)

#print("lr.coef_:", lr.coef_)
#print("lr.intercept_:", lr.intercept_)

print("Training set score: {:.2f}".format(lr.score(xTrain, yTrain)))
print("Test set score: {:.2f}".format(lr.score(xTest, yTest)))
```

```
Training set score: 0.73
Test set score: 0.60
```

Ridge Regression

```
ridge = Ridge().fit(xTrain, yTrain)
print("Training set score: {:.2f}".format(ridge.score(xTrain, yTrain)))
print("Test set score: {:.2f}".format(ridge.score(xTest, yTest)))
```

```
Training set score: 0.79
Test set score: 0.60
```

Decision Trees

```
tree1 = DecisionTreeRegressor(max_depth=5).fit(xTrain, yTrain)
tree2 = DecisionTreeRegressor(max_depth=10).fit(xTrain, yTrain)
print ("Tree of depth 5 Training set score: {:.2f}".format(tree1.score(xTrain, yTrain)))
print ("Tree of depth 5 Test set score: {:.2f}".format(tree1.score(xTest, yTest)))
print ("Tree of depth 10 Training set score: {:.2f}".format(tree2.score(xTrain, yTrain)))
print ("Tree of depth 10 Test set score: {:.2f}".format(tree2.score(xTest, yTest)))
#higher depth means overfitting to the training data and lower accuracy on the testing data
```

```
Tree of depth 5 Training set score: 0.50
Tree of depth 5 Test set score:0.30
Tree of depth 10 Training set score:0.77
Tree of depth 10 Test set score:0.55
```

K-Nearest Neighbors

```
for n in range(2,12):
    # this loop tries number of neighbors from 2 to 11
    knn = KNeighborsRegressor(n_neighbors=n)
    # KNN with 6 neighbors was found to be the most accurate in general
    knn.fit(xTrain, yTrain)
    print ("Training set score with " + str(n) + " neighbors: {:.2f}".format(knn.score(xTrain, yTr
ain)))
    print ("Test set score with " + str(n) + " neighbors: {:.2f}".format(knn.score(xTest, yTest)))
```

```
Training set score with 6 neighbors: 0.39
Test set score with 6 neighbors: 0.18
```

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

Gradient Boosted Regression¶

```
gbr = GradientBoostingRegressor()
gbr.fit(xTrain, yTrain)
print("Training set score: {:.3f}".format(gbr.score(xTrain, yTrain)))
print("Test set score: {:.3f}".format(gbr.score(xTest, yTest)))
```

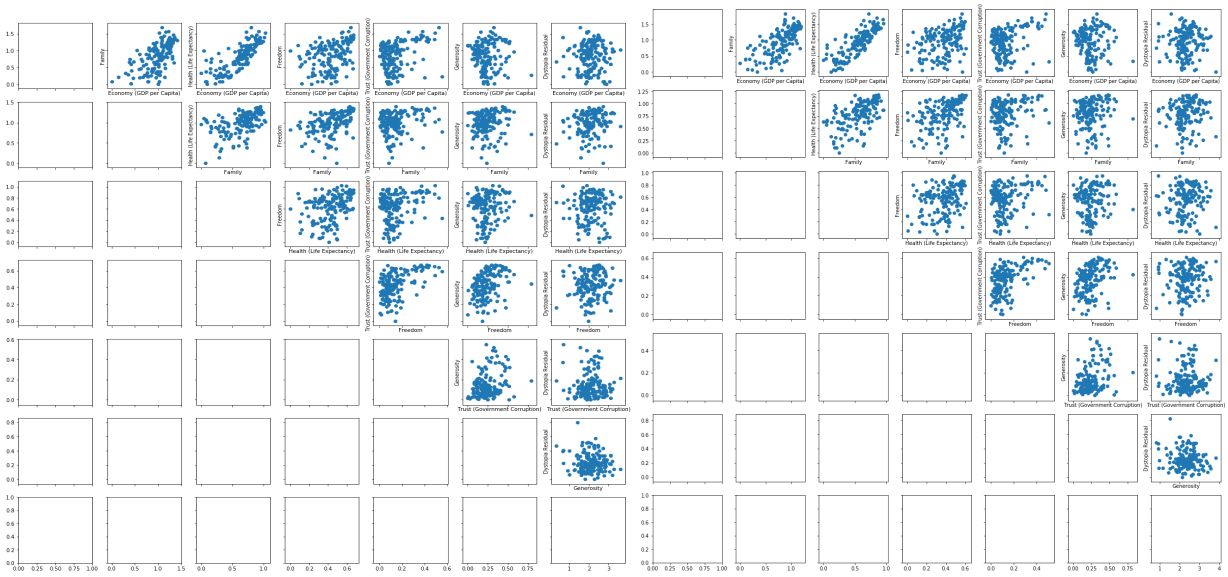
```
Training set score: 0.73
Test set score: 0.50
```

Results and Conclusion:

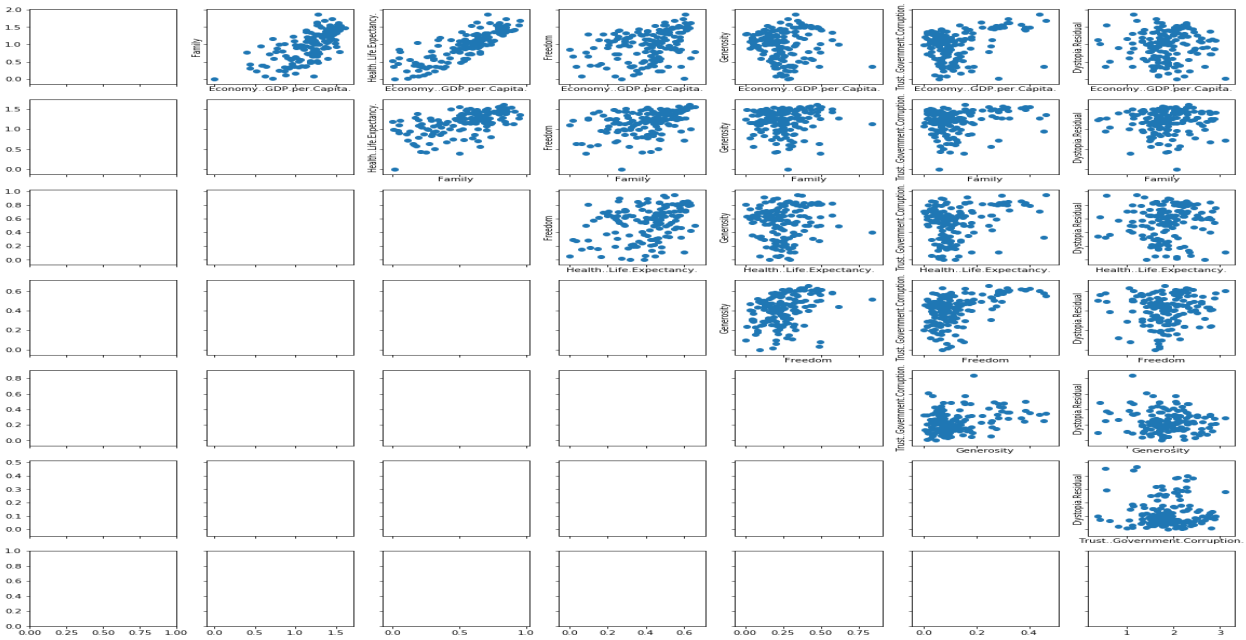
Datastet was imported and data preprocessing has been done by getting rid of all the useless co lumns, and replaced the null values with mode which is most frequent occurrence in the datase t, and then implemented tf-idf(term frequency-inverse document frequency) where I first creat ed vector using only 1000 words and removing useless stop words because we just consider wo rds that matter to the target variable (salary) then preprocess the categorical features and then splitted my data in 80% of training and 20% test set and applied different models in which It see ms the linear and ridge regression the most accurate and best among all in training and test sco re than other models.

Question 1: Happiness

Preliminary Visualization



Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)



PCA Dimension Reduction on data2015, data2016 and data2017

```
#PCA Dimension on 2015 data
pca = PCA().fit(data2015)
ratios = np.cumsum(pca.explained_variance_ratio_)
#ratios = temp/(np.arange(len(temp))+1)
maxRatioIndex = 2
for n in range(3,len(ratios)):
    if ratios[n]/(n/len(ratios)+1) > ratios[maxRatioIndex]/(maxRatioIndex/len(ratios)+1):
        maxRatioIndex = n
#ratios = sorted(ratios)
plt.plot (ratios)
print ("The number of principle components with the highest ratio of variance to components is", maxRatioIndex)
print ("Using", maxRatioIndex, "components will preserve", str(100*ratios[maxRatioIndex].round(4)) + "% of the data
plt.show()
pca = PCA(n_components=maxRatioIndex)
pca.fit(data2015)
data2015 = pca.transform(data2015)
print (data2015.shape)
```

```
The number of principle components with the highest ratio of variance to components is 6
Using 6 components will preserve 88.52% of the data
```

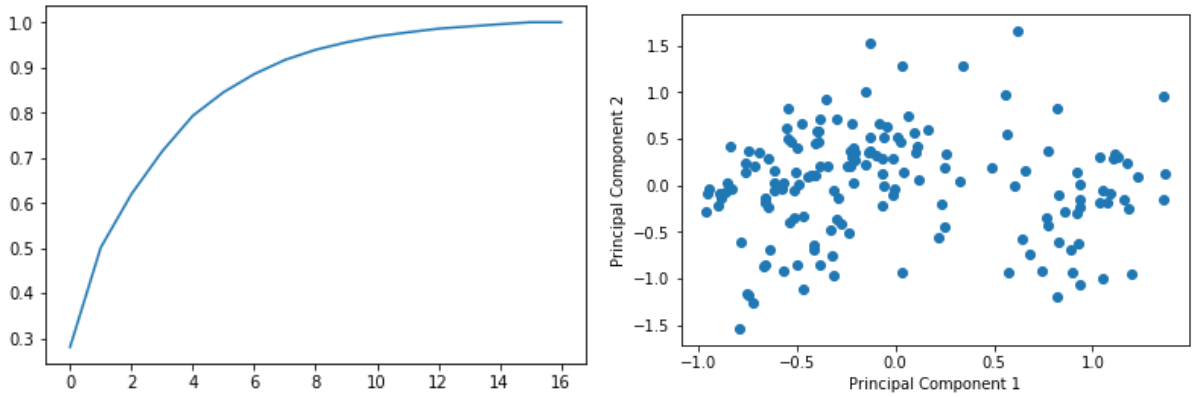


Fig: PCA Dimension reduction on Data2015

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

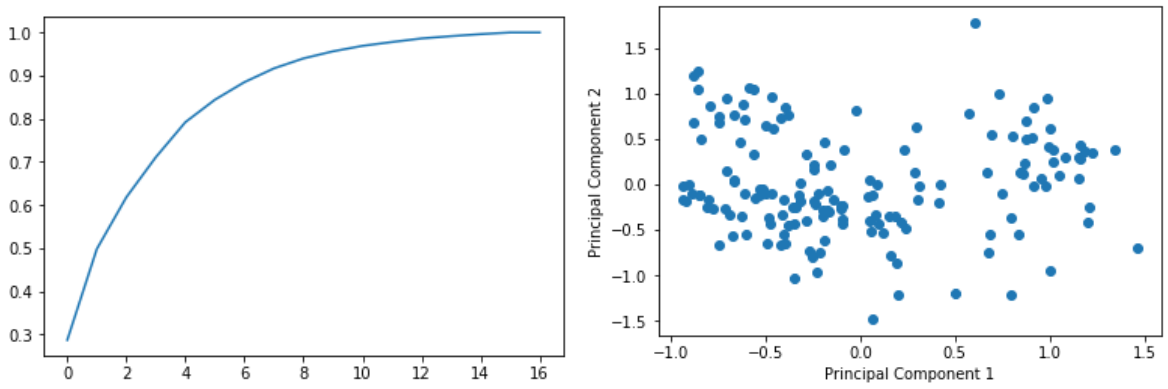


Fig: PCA Dimension reduction on Data2016

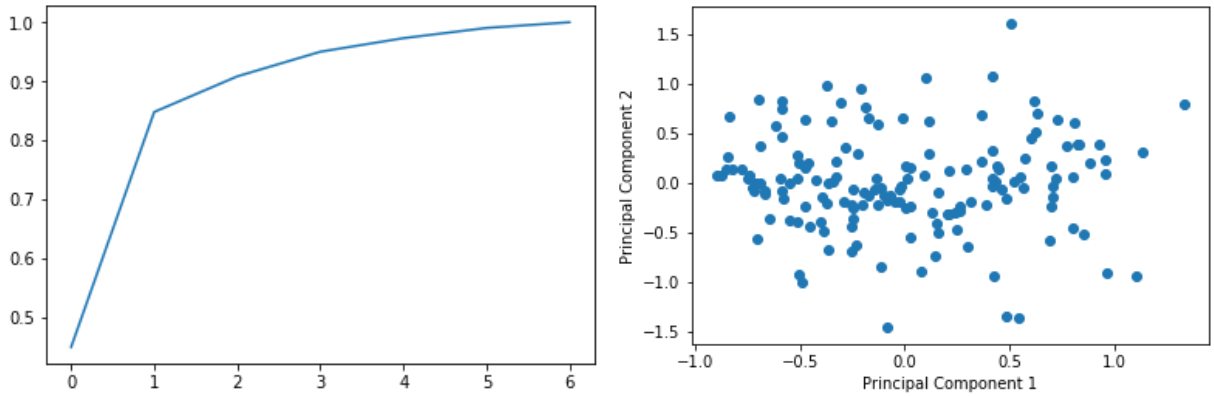


Fig: PCA Dimension reduction on Data2017

KMeans

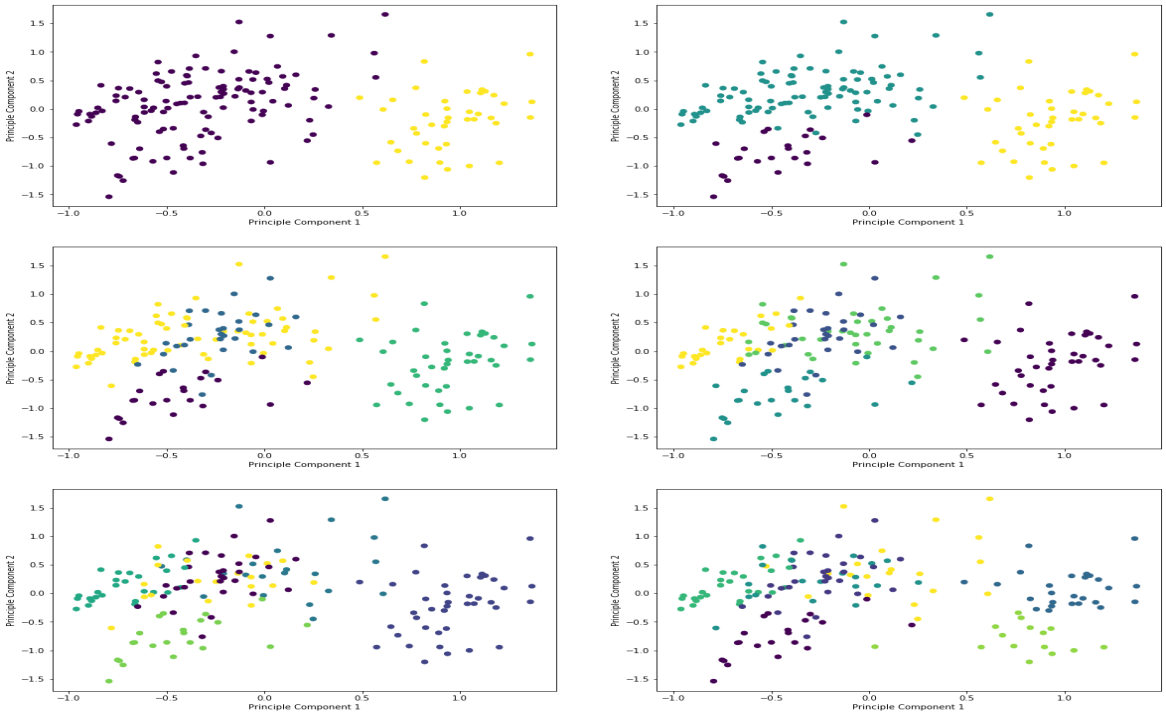


Fig: Kmeans on Data2015

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

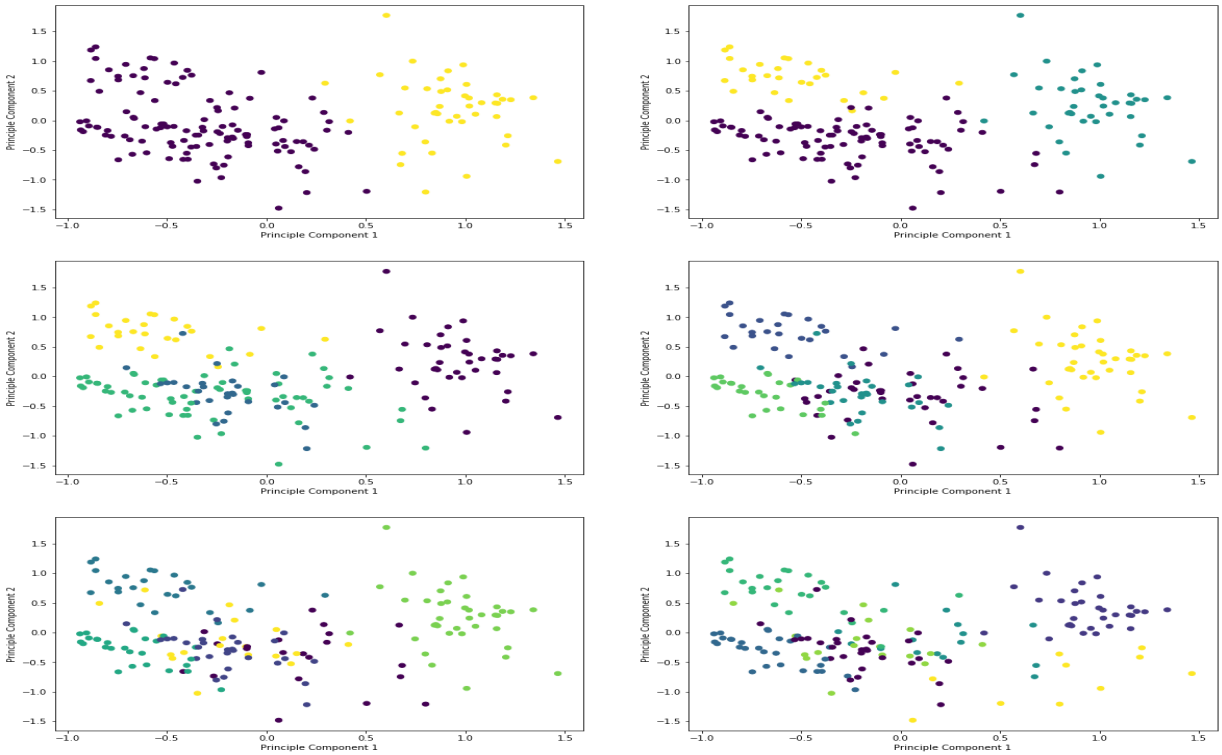


Fig: Kmeans on Data2016

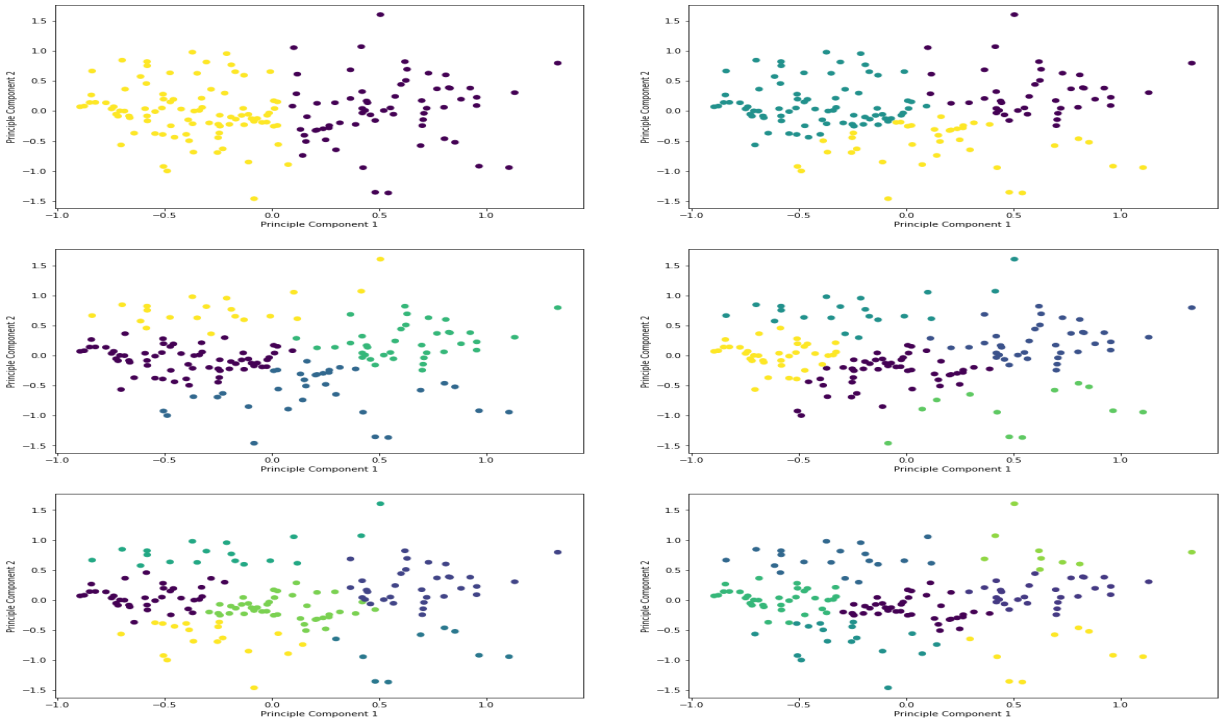


Fig: Kmeans on Data2017

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

GMM

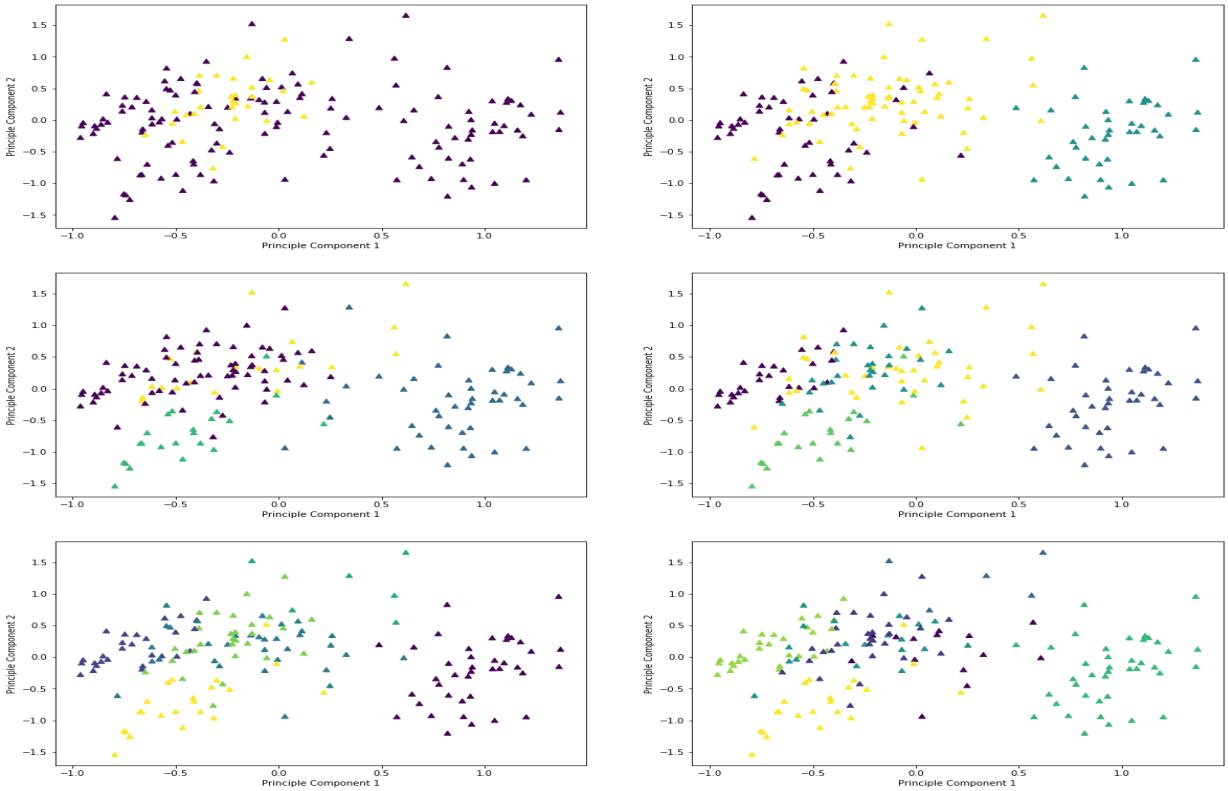


Fig: GMM on Data2015

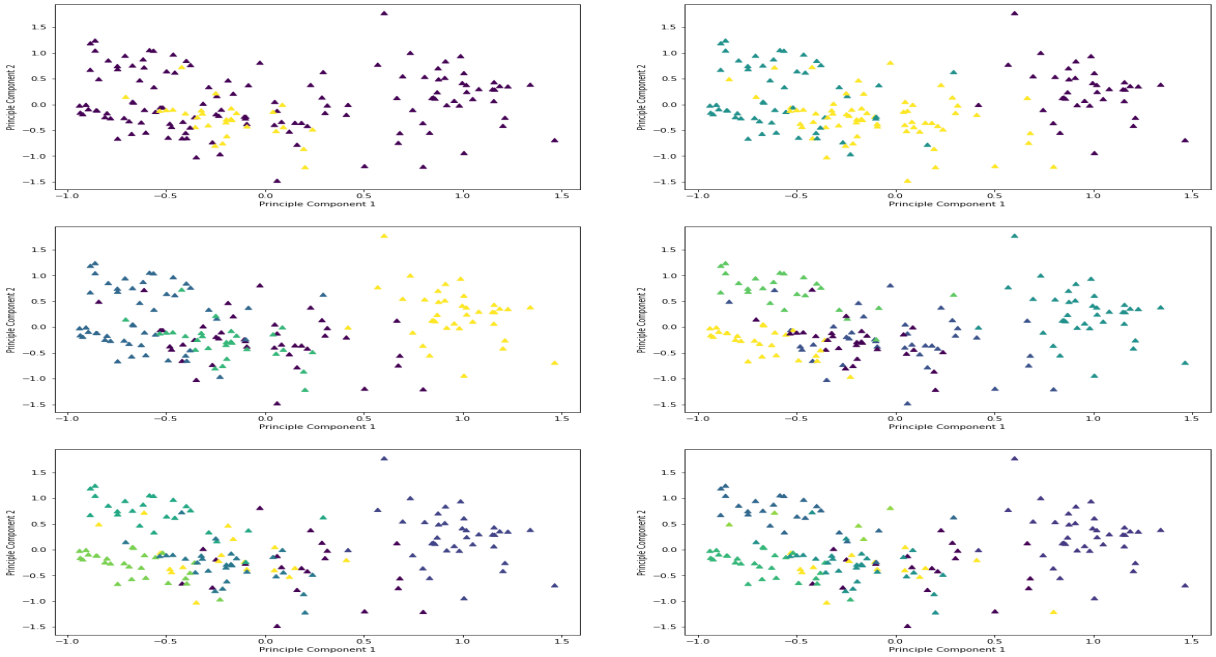


Fig: GMM on Data2016

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

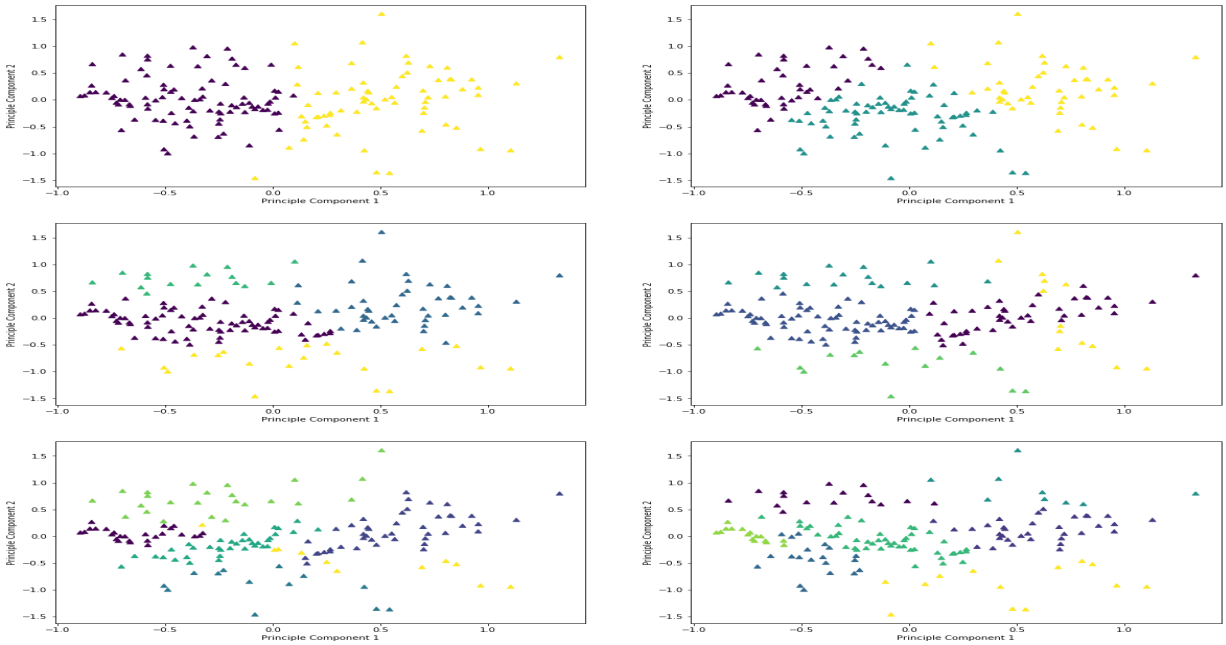


Fig: GMM on Data2017

AgglomerativeClusutering

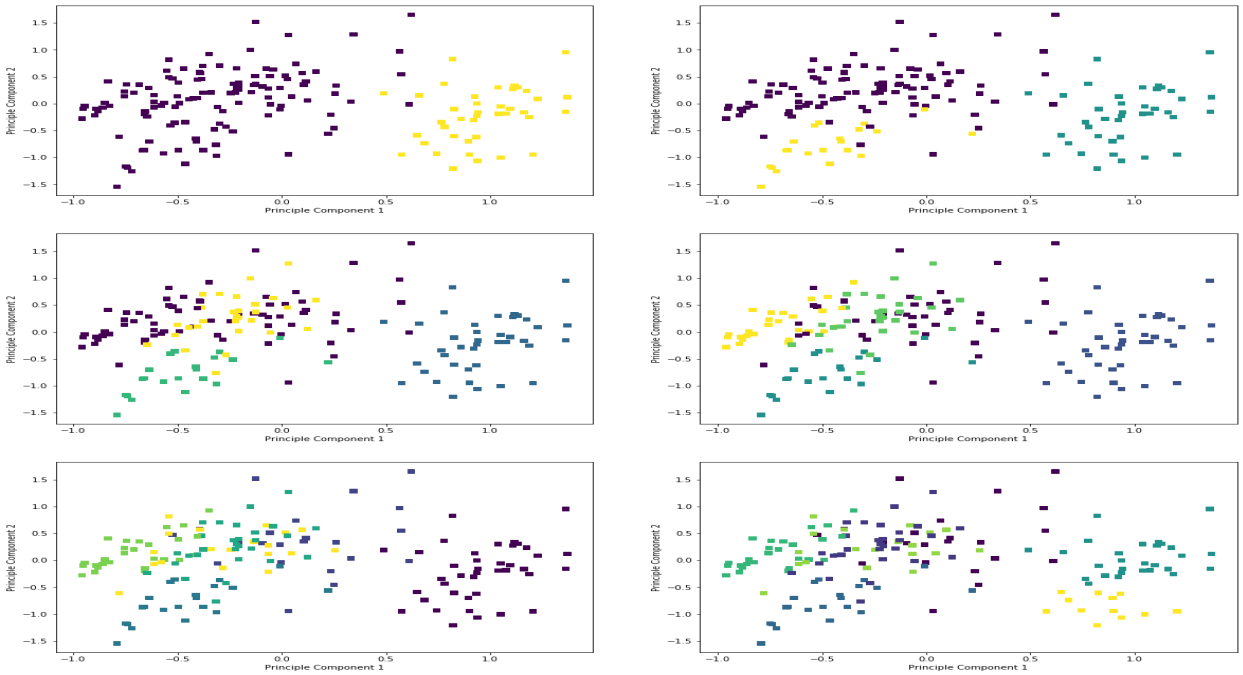


Fig: AgglomerativeClustering on Data2015

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

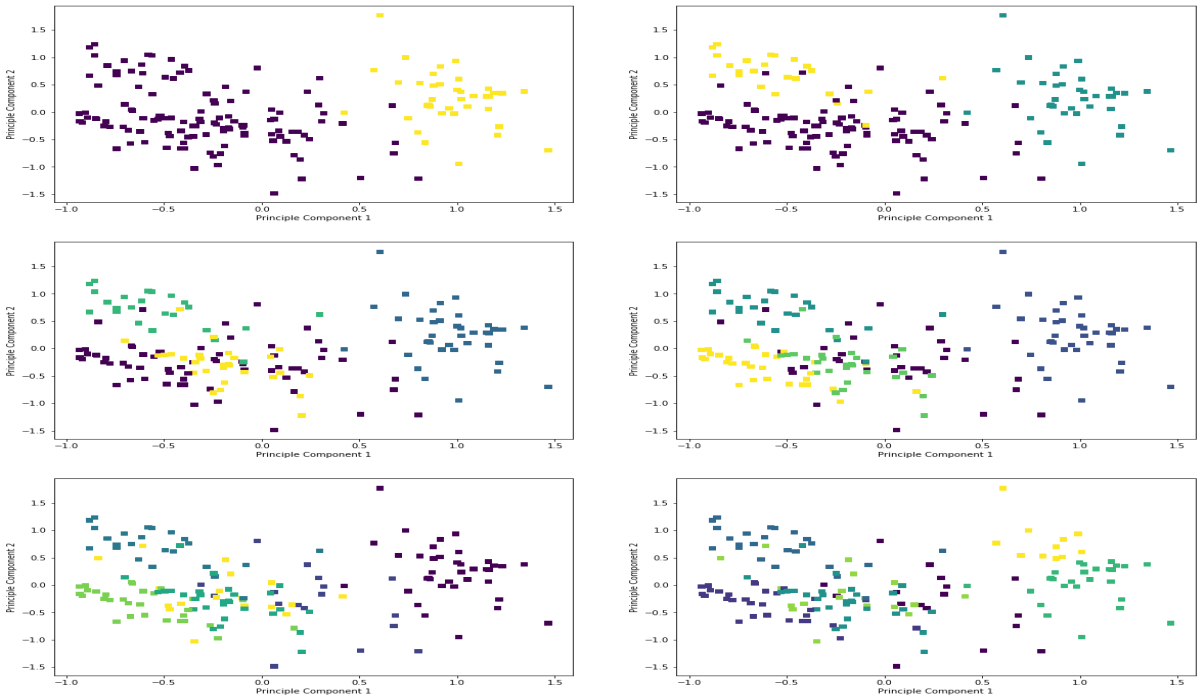


Fig: AgglomerativeClustering on Data2016

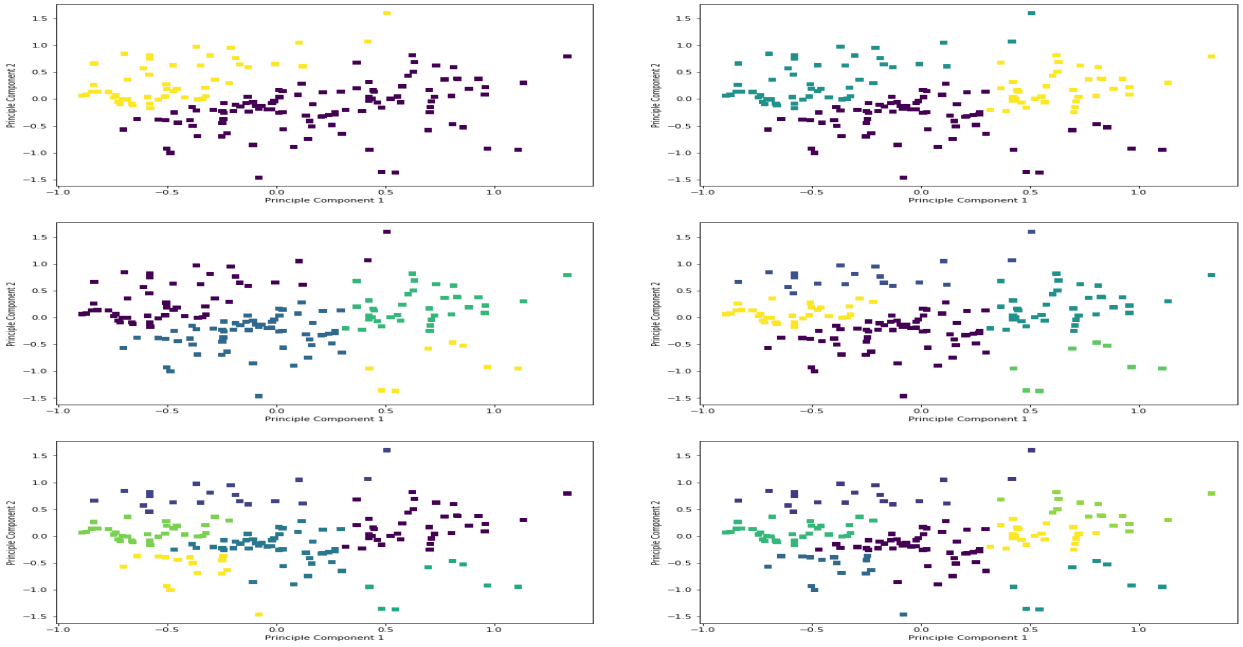


Fig: AgglomerativeClustering on Data2017

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

Spectral Clustering

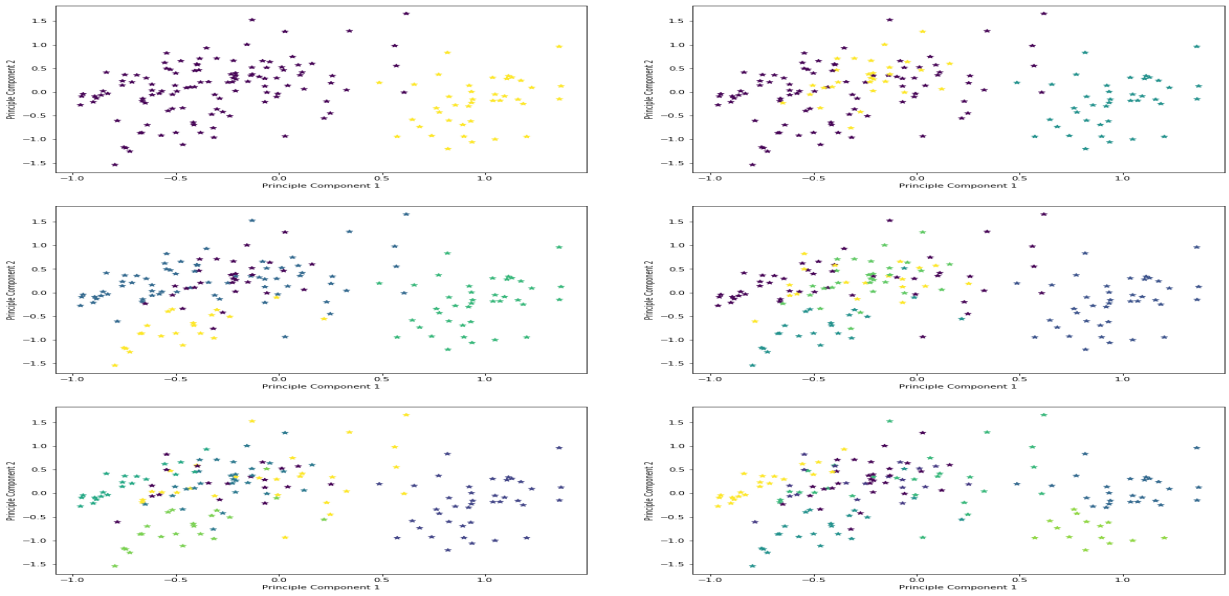


Fig: Spectral Clustering on Data2015

Since we already clustered with numClusters equal to the number of countries, we simply rank the countries by which cluster they are in.

Agglomerative seriation Ranking chart:

```
: 1 data2015original.nlargest(20,'agglo') #nlargest is happiest
```

:

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual	kmeans	gmm	ag
50	Bolivia	Latin America and Caribbean	51	5.890	0.05642	0.68133	0.97841	0.53920	0.57414	0.08800	0.20536	2.82334	123	123	
24	Panama	Latin America and Caribbean	25	6.786	0.04910	1.06353	1.19850	0.79661	0.54210	0.09270	0.24434	2.84848	40	156	
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738	155	155	
40	Trinidad and Tobago	Latin America and Caribbean	41	6.168	0.10895	1.21183	1.18354	0.61483	0.55884	0.01140	0.31844	2.26882	152	153	
8	New Zealand	Australia and New Zealand	9	7.286	0.03371	1.25018	1.31967	0.90837	0.63938	0.42922	0.47501	2.26425	154	12	
31	Uruguay	Latin America and Caribbean	32	6.485	0.04539	1.06166	1.20890	0.81160	0.60362	0.24558	0.23240	2.32142	65	55	
47	Ecuador	Latin America and Caribbean	48	5.975	0.04528	0.86402	0.99903	0.79075	0.48574	0.18090	0.11541	2.53942	1	151	
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.49204	91	150	
6	Netherlands	Western	7	7.378	0.02799	1.32944	1.28017	0.89284	0.61576	0.31814	0.47610	2.46570	148	148	

KMeans Seriation Ranking table:

STAT 517 PROJECT 2

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

```
data2015original.nlargest(20,'kmeans')
```

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual	kmeans	gmm	a
148	Chad	Sub-Saharan Africa	149	3.667	0.03830	0.34193	0.76062	0.15010	0.23501	0.05269	0.18386	1.94296	157	68	
29	Argentina	Latin America and Caribbean	30	6.574	0.04612	1.05351	1.24823	0.78723	0.44974	0.08484	0.11451	2.83600	156	0	
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738	155	155	
8	New Zealand	Australia and New Zealand	9	7.286	0.03371	1.25018	1.31967	0.90837	0.63938	0.42922	0.47501	2.26425	154	12	
79	Azerbaijan	Central and Eastern Europe	80	5.212	0.03363	1.02389	0.93793	0.64045	0.37030	0.16065	0.07799	2.00073	153	154	
40	Trinidad and Tobago	Latin America and Caribbean	41	6.168	0.10895	1.21183	1.18354	0.61483	0.55884	0.01140	0.31844	2.26882	152	153	
12	Austria	Western Europe	13	7.200	0.03751	1.33723	1.29704	0.89042	0.62433	0.18676	0.33088	2.53320	151	27	
57	Peru	Latin America and Caribbean	58	5.824	0.04615	0.90019	0.97459	0.73017	0.41496	0.05989	0.14982	2.59450	150	67	

1) Norway tops the global happiness rankings for all three years 2015, 2016, and 2017.
Answer: Norway tops in the original dataset of happiness ranking but Bolivia, serbia and vietnam tops the ranking according to seriation analysis of data2015, data2016, data2017 respectively. My clustering have found Chad, serbia and south sudan tops the Kmeans seriation in data2015, data2016 and data2017 respectively.

2) All top ten countries rank highly on all the main features found to support happiness.
Answer: It seems top ten countries indeed having the high rankings and scores in some features but low in some features like Kmeans seriation score for instance Kmeans of panama and Ecuador is very low in numbers of 40 and 1 respectively.

3) Happiness is both social and personal.
Answer: Yes Happiness founds to be both social and personal as all factors involved which influence happiness by ecomic and government.

4) Unemployment causes a major fall in happiness, and even for those in work the quality of work can cause major variations in happiness
Answer: This is true indeed unemployment caused a major fall in happiness also other factors mattered and happiness get influenced by those like economy and health.

5) China are no happier than most countries, though richer and longer longevity Answer: China did not appeared anywhere in the happiness ranking which seems as no more the happier country as I found Serbia being the most happiest according to both seriation analysis.

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

6) Much of Africa is struggling
Answer: Sub-saharan Africa and central african republic seems to be the most saddest in all analysis of years

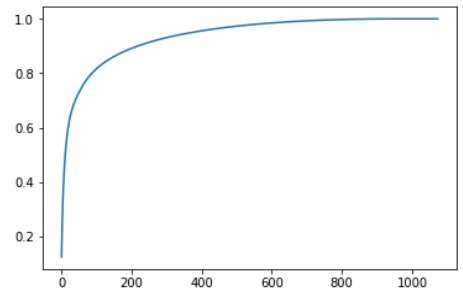
7) Happiness has fallen in America
Answer: It seems America did not topped but is not lowest as well It pretty in middle and average in happiness regions of all years.

Question 2: Mitochondria

PCA Dimension Reduction

```
pca = PCA().fit(data)
ratios = np.cumsum(pca.explained_variance_ratio_)
#ratios = temp/(np.arange(len(temp))+1)
maxRatioIndex = 0
for n in range(1,len(ratios)):
    if ratios[n]/(n/len(ratios)+1) > ratios[maxRatioIndex]/(maxRatioIndex/len(ratios)+1):
        maxRatioIndex = n
#ratios = sorted(ratios)
plt.plot (ratios)
print ("The number of principle components with the highest ratio of variance to components is", maxRatioIndex)
print ("Using", maxRatioIndex, "components will preserve", str(100*ratios[maxRatioIndex].round(4)) + "% of the data
plt.show()
pca = PCA(n_components=maxRatioIndex)
pca.fit(data)
data = pca.transform(data)
print (data.shape)
```

```
The number of principle components with the highest ratio of variance to components is 150
Using 150 components will preserve 86.18% of the data
```




```
(1074, 150)
```

KMeans

```
# first find optimal number of clusters using silhouette scores
silhouetteScores = []
for numClusters in range(3,50):
    kmeans = KMeans(n_clusters=numClusters, random_state=int(time.time()))
    pred = kmeans.fit_predict(data)
    silhouetteScores.append([silhouette_score(data, pred),numClusters])
optimal = sorted(silhouetteScores)[-1][1]
print ("The optimal number of clusters according to silhouette scores is", optimal)
```

```
The optimal number of clusters according to silhouette scores is 40
```

```
plt.figure(figsize=(10,10))
kmeans = KMeans(n_clusters=optimal, random_state=int(time.time()))
kmeans.fit(data)
kmeansLabels = kmeans.predict(data)
plt.xlabel('Principle Component 1')
plt.ylabel('Principle Component 2')
plt.scatter(data[:, 0], data[:, 1], c=kmeansLabels, marker="o", s=40, cmap='viridis')
plt.show()
```

STAT 517 PROJECT 2

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

GMM

```
# first find optimal number of clusters using silhouette scores
silhouetteScores = []
for numComponents in range(3,50):
    gmm = GaussianMixture(n_components=numComponents, covariance_type='full')
    gmm.fit(data)
    pred = gmm.predict(data)
    silhouetteScores.append([silhouette_score(data, pred),numClusters])
optimal = sorted(silhouetteScores)[-1][1]
print ("The optimal number of clusters according to silhouette scores is", optimal)
```

```
The optimal number of clusters according to silhouette scores is 49
```

```
plt.figure(figsize=(10,10))
gmm = GaussianMixture(n_components=optimal, covariance_type='full').fit(data)
gmmLabels = gmm.predict(data)
plt.xlabel('Principle Component 1')
plt.ylabel('Principle Component 2')
plt.scatter(data[:, 0], data[:, 1], c=gmmLabels, marker="^", s=40, cmap='viridis')
plt.show()
```

AgglomerativeClusutering

```
# first find optimal number of clusters using silhouette scores
silhouetteScores = []
for numClusters in range(3,50):
    agglo = AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto', connectivity=None, linkage='wa
    pred = agglo.fit_predict(data)
    silhouetteScores.append([silhouette_score(data, pred),numClusters])
optimal = sorted(silhouetteScores)[-1][1]
print ("The optimal number of clusters according to silhouette scores is", optimal)
```

```
The optimal number of clusters according to silhouette scores is 39
```

```
plt.figure(figsize=(10,10))
agglo = AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto', connectivity=None, linkage='ward',
agglo.fit(data)
aggloLabels = agglo.labels_
plt.xlabel('Principle Component 1')
plt.ylabel('Principle Component 2')
plt.scatter(data[:, 0], data[:, 1], c=aggloLabels, marker="s", s=40, cmap='viridis')
plt.show()
```

Spectral Clustering

```
# first find optimal number of clusters using silhouette scores
silhouetteScores = []
for numClusters in range(3,50):
    spectral = SpectralClustering(n_clusters=numClusters, affinity='nearest_neighbors', assign_labels='kmeans')
    pred = spectral.fit_predict(data)
    silhouetteScores.append([silhouette_score(data, pred),numClusters])
optimal = sorted(silhouetteScores)[-1][1]
print ("The optimal number of clusters according to silhouette scores is", optimal)
```

```
/Users/rohit1/anaconda3/lib/python3.6/site-packages/sklearn/manifold/spectral_embedding_.py:234: UserWarning: Graph i
s not fully connected, spectral embedding may not work as expected.
  warnings.warn("Graph is not fully connected, spectral embedding"
```

```
The optimal number of clusters according to silhouette scores is 39
```

```
plt.figure(figsize=(10,10))
spectral = SpectralClustering(n_clusters=optimal, affinity='nearest_neighbors', assign_labels='kmeans')
spectral.fit(data)
spectralLabels = spectral.labels_
plt.xlabel('Principle Component 1')
plt.ylabel('Principle Component 2')
plt.scatter(data[:, 0], data[:, 1], c=spectralLabels, marker="*", s=40, cmap='viridis')
plt.show()
```

```
/Users/rohit1/anaconda3/lib/python3.6/site-packages/sklearn/manifold/spectral_embedding_.py:234: UserWarning: Graph i
s not fully connected, spectral embedding may not work as expected.
  warnings.warn("Graph is not fully connected, spectral embedding"
```

STAT 517 PROJECT 2

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

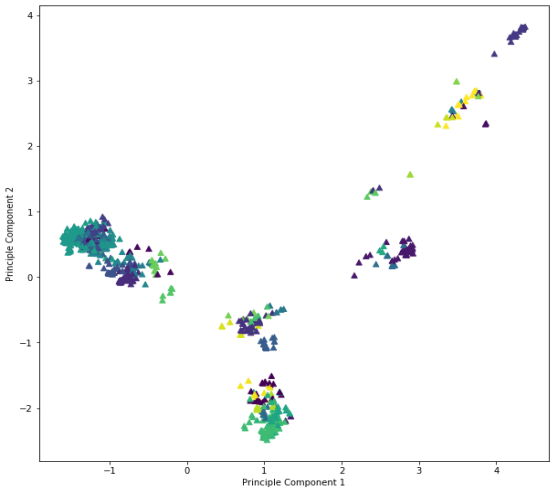


Fig: KMeans

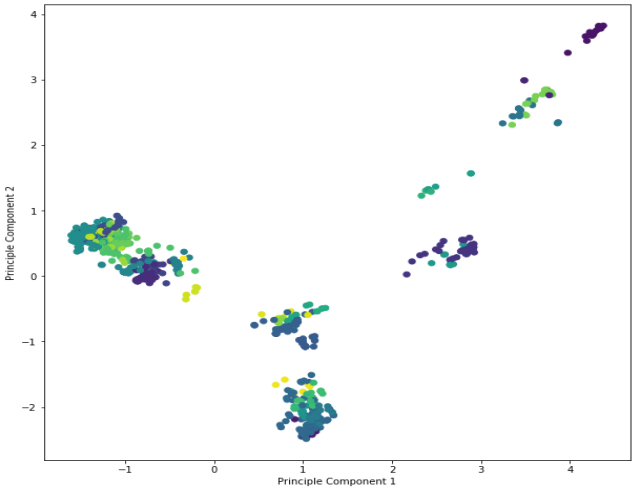


Fig: GMM

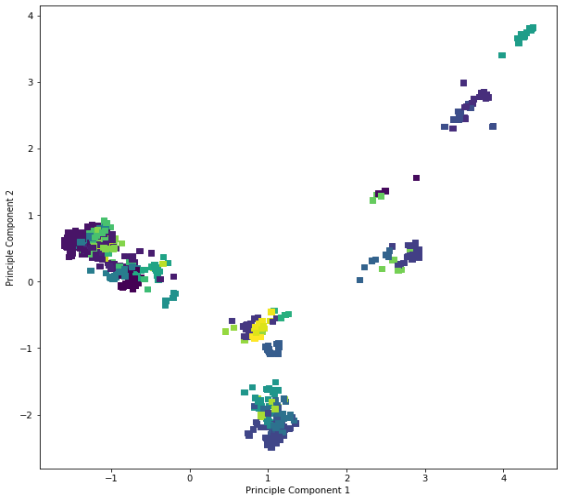


Fig: agglomerative clustering.

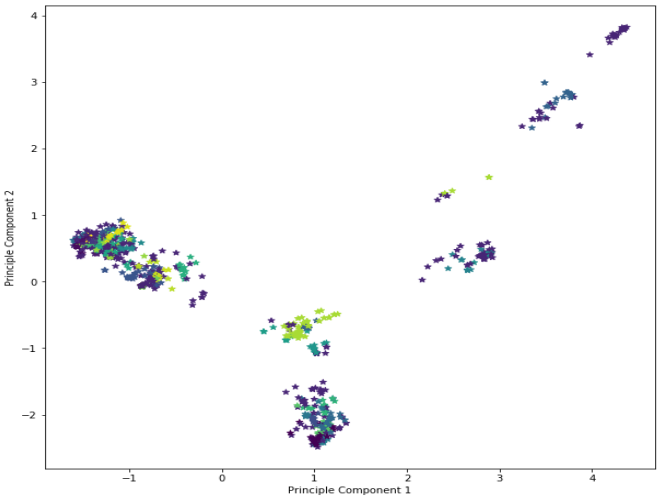
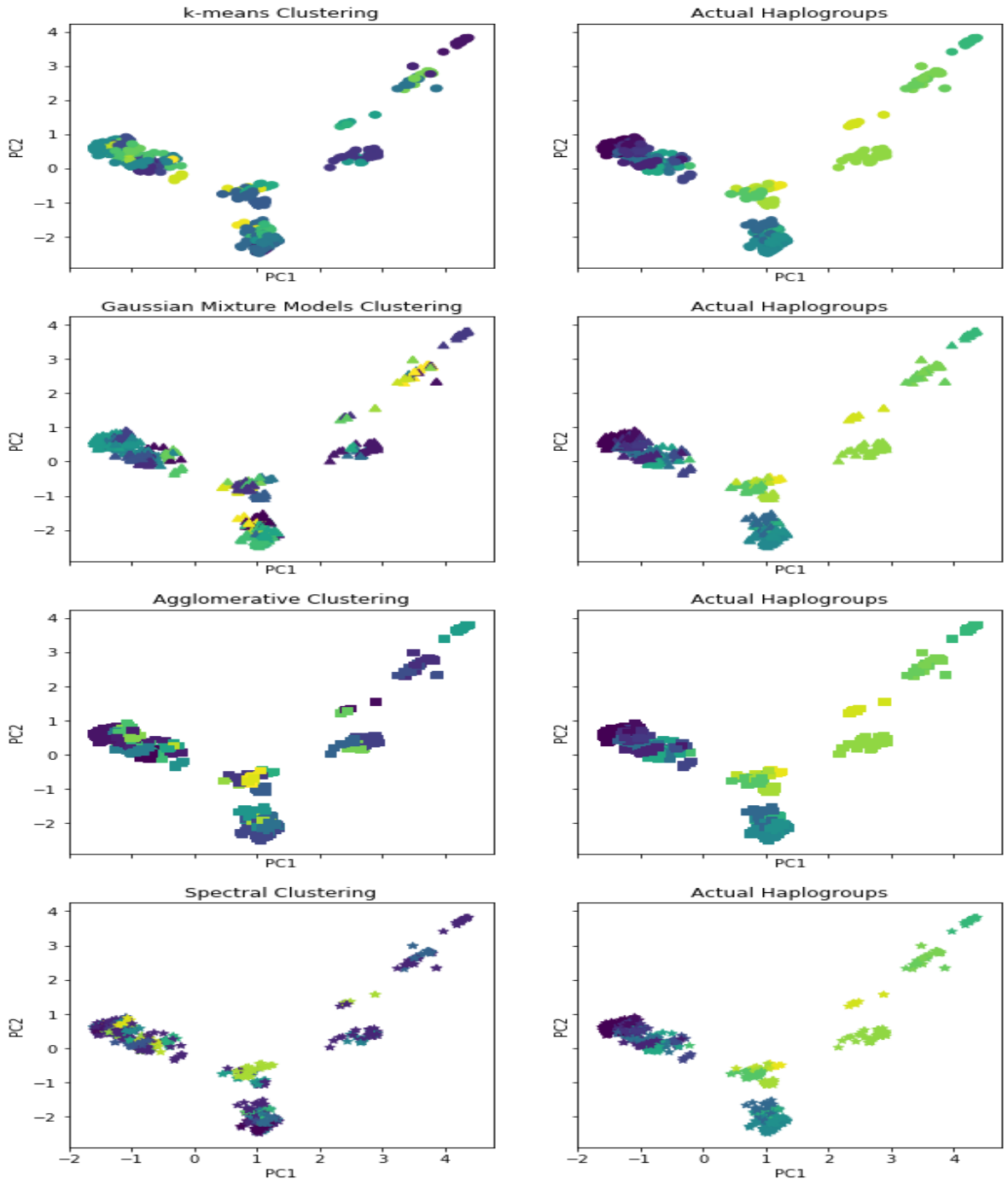


Fig: Spectral clustering

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)



This shows that although the clustering algorithms sometimes found more clusters than haplogroups, they are finding similar enough clusters to be considered accurate.

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

Question 3: Data Mining the Bible

Preliminary Visualization

Word cloud is installed in python for this representation of 66 different most frequent words are reprsented which appeared in both the testaments

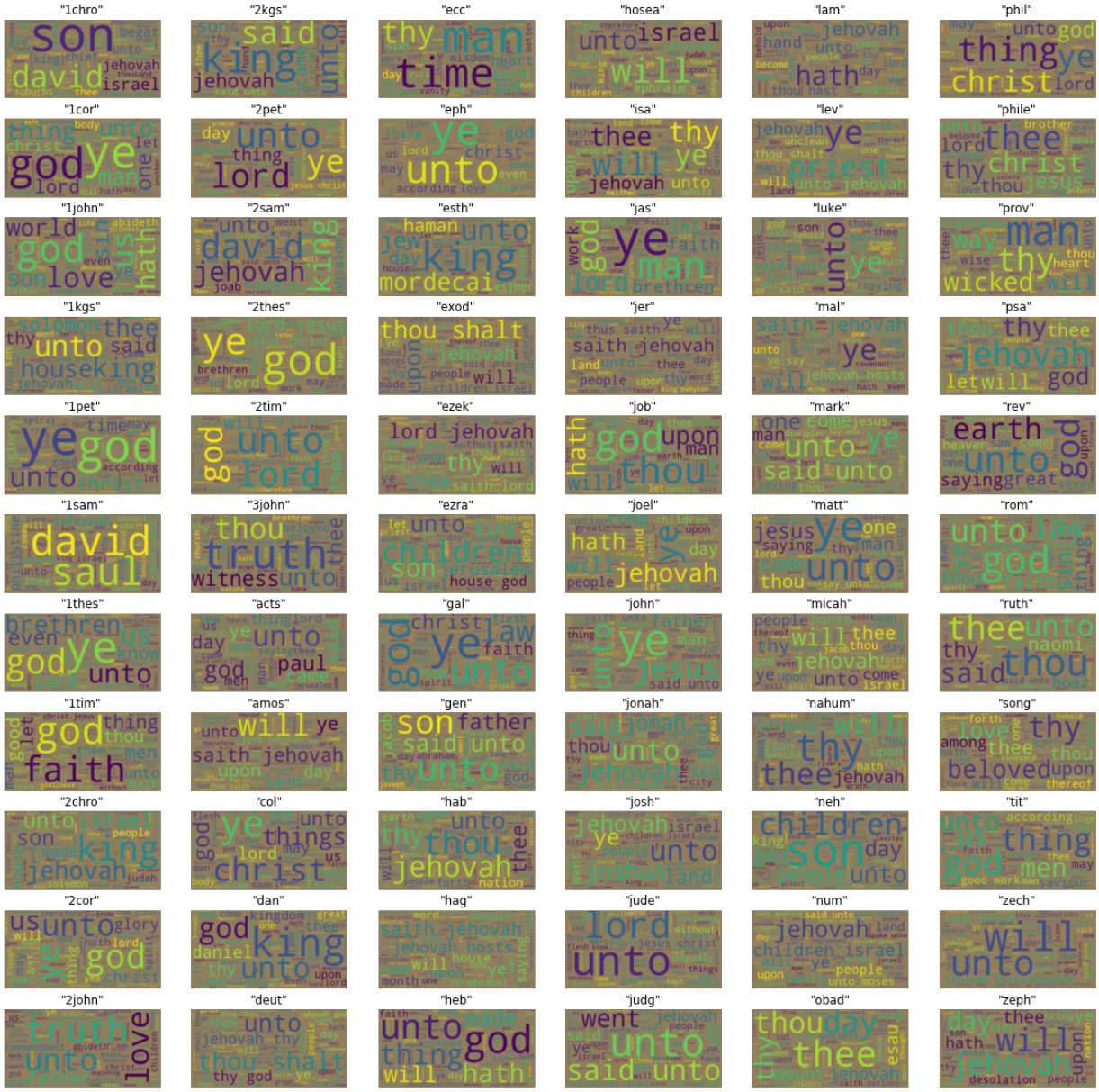


Fig: 66 word cloud

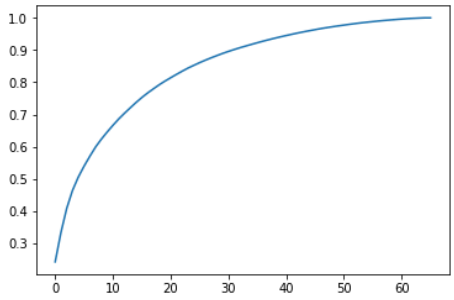
STAT 517 PROJECT 2

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

PCA Dimension Reduction

```
pca = PCA().fit(data)
ratios = np.cumsum(pca.explained_variance_ratio_)
#ratios = temp/(np.arange(len(temp))+1)
maxRatioIndex = 0
for n in range(1,len(ratios)):
    if ratios[n]/(n/len(ratios)+1) > ratios[maxRatioIndex]/(maxRatioIndex/len(ratios)+1):
        maxRatioIndex = n
#ratios = sorted(ratios)
plt.plot (ratios)
print ("The number of principle components with the highest ratio of variance to components is", maxRatioIndex)
print ("Using", maxRatioIndex, "components will preserve", str(100*ratios[maxRatioIndex].round(4)) + "% of the data
plt.show()
pca = PCA(n_components=maxRatioIndex)
pca.fit(data)
dataPCA = pca.transform(data)
print (dataPCA.shape)
```

```
The number of principle components with the highest ratio of variance to components is 23
Using 23 components will preserve 84.41% of the data
```




```
(66, 23)
```

KMeans Clustering

```
# first find optimal number of clusters using silhouette scores
silhouetteScores = []
for numClusters in range(3,50):
    kmeans = KMeans(n_clusters=numClusters, random_state=int(time.time()))
    pred = kmeans.fit_predict(dataPCA)
    silhouetteScores.append([silhouette_score(dataPCA, pred),numClusters])
kmeansOptimal = sorted(silhouetteScores)[-1][1]
print ("The optimal number of clusters according to silhouette scores is", optimal)
plt.figure(figsize=(10,10))
kmeans = KMeans(n_clusters=kmeansOptimal, random_state=int(time.time())).fit(dataPCA)
kmeansLabels = kmeans.predict(dataPCA)
kmeans = KMeans(n_clusters=2, random_state=int(time.time())).fit(dataPCA)
kmeansLabels2 = kmeans.predict(dataPCA)
```

```
The optimal number of clusters according to silhouette scores is 3
```

```
<Figure size 720x720 with 0 Axes>
```

DBSCAN Clustering

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

```
# first find optimal number of clusters using silhouette scores
silhouetteScores = []
for numClusters in range(3,50):
    db = DBSCAN().fit(dataPCA)
    pred = db.labels_
    silhouetteScores.append([silhouette_score(dataPCA, pred),numClusters])
optimal = sorted(silhouetteScores)[-1][1]
print ("The optimal number of clusters according to silhouette scores is", optimal)
plt.figure(figsize=(10,10))
db = DBSCAN().fit(dataPCA)
dbLabels = db.labels_
```

```
The optimal number of clusters according to silhouette scores is 49

<Figure size 720x720 with 0 Axes>
```

Agglomerative Clustering

```
# first find optimal number of clusters using silhouette scores
silhouetteScores = []
for numClusters in range(3,50):
    agglo = AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto', connectivity=None, linkage='wa
    pred = agglo.fit_predict(dataPCA)
    silhouetteScores.append([silhouette_score(dataPCA, pred),numClusters])
aggloOptimal = sorted(silhouetteScores)[-1][1]
print ("The optimal number of clusters according to silhouette scores is", optimal)
plt.figure(figsize=(10,10))
agglo = AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto', connectivity=None, linkage='ward',
aggloLabels = agglo.labels_
agglo = AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto', connectivity=None, linkage='ward',
aggloLabels2 = agglo.labels_
```

```
The optimal number of clusters according to silhouette scores is 49

<Figure size 720x720 with 0 Axes>
```

Spectral Clustering

```
# first find optimal number of clusters using silhouette scores
silhouetteScores = []
for numClusters in range(3,50):
    spectral = SpectralClustering(n_clusters=numClusters, affinity='nearest_neighbors', assign_labels='kmeans')
    pred = spectral.fit_predict(dataPCA)
    silhouetteScores.append([silhouette_score(data, pred),numClusters])
spectralOptimal = sorted(silhouetteScores)[-1][1]
print ("The optimal number of clusters according to silhouette scores is", optimal)
plt.figure(figsize=(10,10))
spectral = SpectralClustering(n_clusters=spectralOptimal, affinity='nearest_neighbors', assign_labels='kmeans').fit
spectralLabels = spectral.labels_
spectral = SpectralClustering(n_clusters=2, affinity='nearest_neighbors', assign_labels='kmeans').fit(dataPCA)
spectralLabels2 = spectral.labels_
```

```
The optimal number of clusters according to silhouette scores is 49

<Figure size 720x720 with 0 Axes>
```

STAT 517 PROJECT 2

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

Cluster Visualization

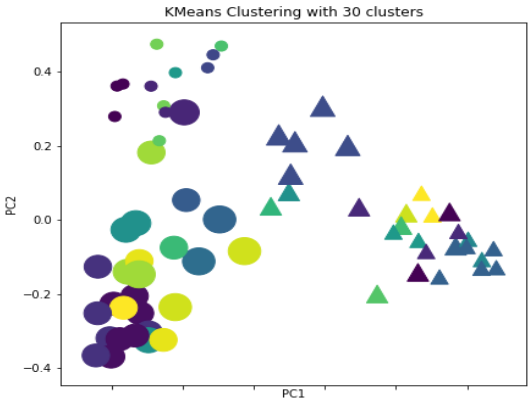
```
sections = {m:(n*100+100) for n, m in enumerate(biblebooks['Sections'].unique())}
bibledata['kmeansLabels'] = kmeansLabels
bibledata['kmeansLabels2'] = kmeansLabels2
bibledata['dbLabels'] = dbLabels
bibledata['aggloLabels'] = aggloLabels
bibledata['aggloLabels2'] = aggloLabels2
bibledata['spectralLabels'] = spectralLabels
bibledata['spectralLabels2'] = spectralLabels2
cat = []
for n, row in biblebooks.iterrows():
    cat.append(row['Testaments'] == 'OT')
bibledataOT = bibledata.loc[cat]
bibledataNT = bibledata.loc[[not b for b in cat]]
```

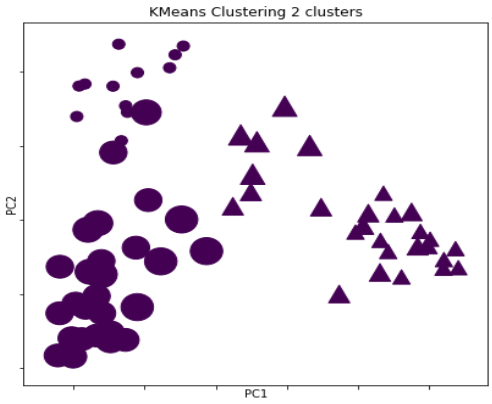
```
dataPCAot = []
dataPCAnt = []
for n, row in enumerate(dataPCA):
    if cat[n]:
        dataPCAot.append(row.tolist())
    else:
        dataPCAnt.append(row.tolist())
dataPCAot = np.array(dataPCAot)
dataPCAnt = np.array(dataPCAnt)
print (dataPCAot[:,0])
#dataPCAot.append()
#dataPCAnt.append()
```

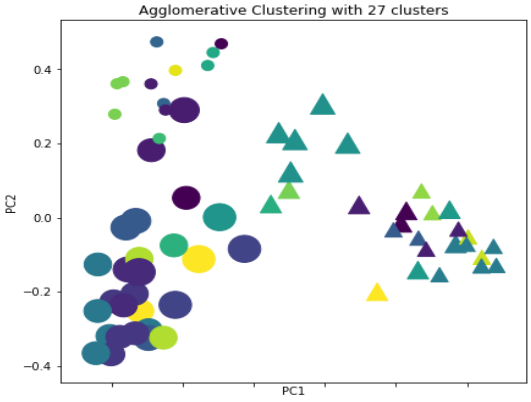
```
[-0.2204052  -0.36812313 -0.25345078 -0.39093024 -0.3844595  -0.27288413
 -0.4052321  -0.18998666 -0.35336041 -0.0261151  -0.091264   -0.35951996
 -0.39425426 -0.12968629 -0.19564554 -0.32009854 -0.3220272  -0.33534226
 -0.40144036 -0.43845658 -0.15475442 -0.29569417 -0.28813132 -0.26574108
 -0.28901246 -0.22462511 -0.2962628  -0.25427615 -0.37703264 -0.36632662
 -0.11432022 -0.33186227 -0.33386044 -0.22083272 -0.32280445 -0.24831119
 -0.09616115 -0.43922101 -0.44413487]
```

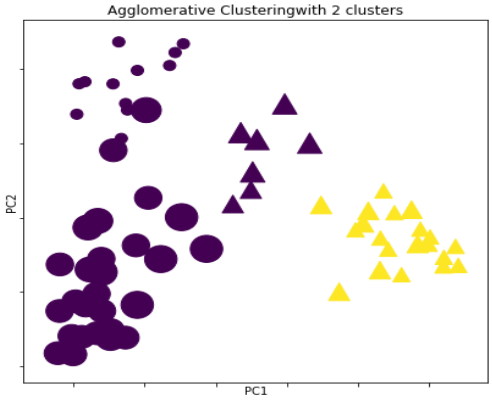
STAT 517 PROJECT 2

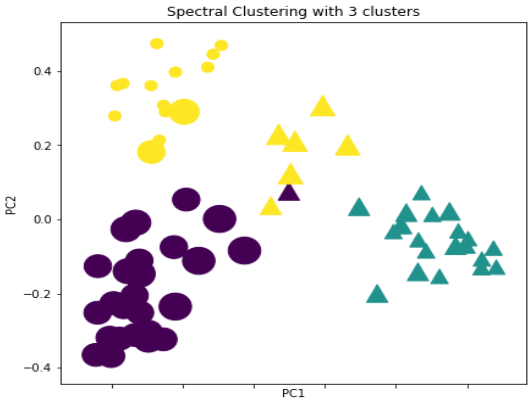
Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

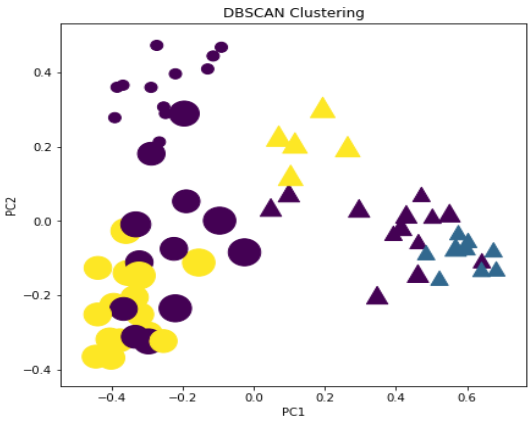












STAT 517 PROJECT 2

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)

Results and Discussions:

A. What is the optimal number of clusters of these 66 Books? Find these clusters and describe them. Are you surprised at your finding? Why/Why not? Graph and color your clusters (probably on the first two PC's). On the graph, show your clusters in colors, the Testaments in plot symbols, and the Sections in sizes.

Answer A: Kmeans with optimal number of clusters 3 found to be more reliable in testing. yes I was surprised by findings as I thought that there must be 66 different clusters but It isn't.

B. How would Association Analyses help to reveal characteristic word clusters? Produce word clouds for the top 10 words clusters with the top 100 most frquent words. Describe these word clusters, and what they are telling you about the Bible. How do these top 10 words clouds represent the 2 Testaments and the 7 Sections?

Answer B: It will help by comparing the most frequently words occured together and from which clusters they belong as for example from the word cloud which is generated having top 100 words from each section and each testaments which consists of most frequently word used in old and new testaments like son,david, king etc.

C. How would Seriation Analyses help to reveal the structure of these 66 Books?

Seriation will put the books in order based on patters in the data. I don't think the books would be put in the same order as in the bible, because this order is fairly arbitrary. It's more likely that the seriation analyses will put the books in order of complexity, or writing style.

STAT 517 PROJECT 2

Yadav, Rohit Kumar (yada6101@vandals.uidaho.edu)