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

Climate Analysis and Temperature Predictions

Abstract

Climate change has been one of the greatest challenge in environment and It poses real and present danger to the future of the world. Also at stake is practically every major life form's existence that resides on this planet. Primarily, addressing climate change requires developing awareness in peoples' minds**.** This Machine Learning Project is based on climate datasets. climate statistics and forecast is an important resource that the government has not explored commensurate to its impact.

The aim of this project is to make this process computerized by implementing principles of data mining and analytics by implementing various types of regression models and clustering. More specifically, this project aims at targeting the social issue of climate condition, analysing data based on climate and temperature, amount of rainfall, snow, thunder and similar factors for different regions in New Delhi, India.

Data can be mined and analysed to find various trends and relations, such as – comparison between climate conditions in area; and can try to predict the temperature on the basis of climate condition in different regions using regression. Clustering, Association analysis, classification and ROC can be used to find features where the climate conditions are mostly similar.

The end result of the project will be research based reports specifying these trends, studied and analysed from data taken over the past few years in New Delhi region on the dataset of 100990 of rows and 20 columns.

FINAL PROJECT STAT 517

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

Background

It is estimated that climate vulenreability will displace 250 million people by 2050 [1]. A more shocking fact is that if everyone in the world lived the way people do in the U.S, it would take five Earths to provide enough resources for everyone [2]. Additionally, thirty seven percent of Americans believe that global warming is a hoax, and 64 percent don't believe that climate change will seriously affect their way of life. [3] The greater problem is how nations choose to tackle this problem of climate change. There have been around 2,950,000 publications on climate change according to Google Scholar. Although most of them produce interesting results, a majority of them fail to attract attention of the general populace through simple media like interactive visualizations. We intend to achieve a part of this function through the present experiment.

Introduction and Research Goals

Climate change and its impact on lives of human has become one of our society's greatest challenges. Data science has had little impact on furthering our understanding of our planet. A lot remains to be understood about our planet and the physical processes that govern it to effectively answer questions about global climate change and its societal impacts. It has been proposed that given the abundance of climate data from different regression models observations and different analysis, we may close some of these knowledge gaps by directly learning from these large climate datasets and can save our environnment from unpredictable weather change. The target is to develop a prediction model for predicting the weather. We can try something like predicting the temperature, visibility, conditions etc. I have done clustering, association, ROC, classification analysis on the similar climate conditions and implemented different types of regression models contributing to climate change, and produced several visualizations which establish relations between several socio-economic factors. More generally, we have created visualizations to trigger climate-aware thought-processes and provides some useful insights through basic statistical and machine learning methods.

The project aims at performing a thorough analysis of Climate Change and temperature datasets made available by the wunderground found on the Kaggle. We found many relations between

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

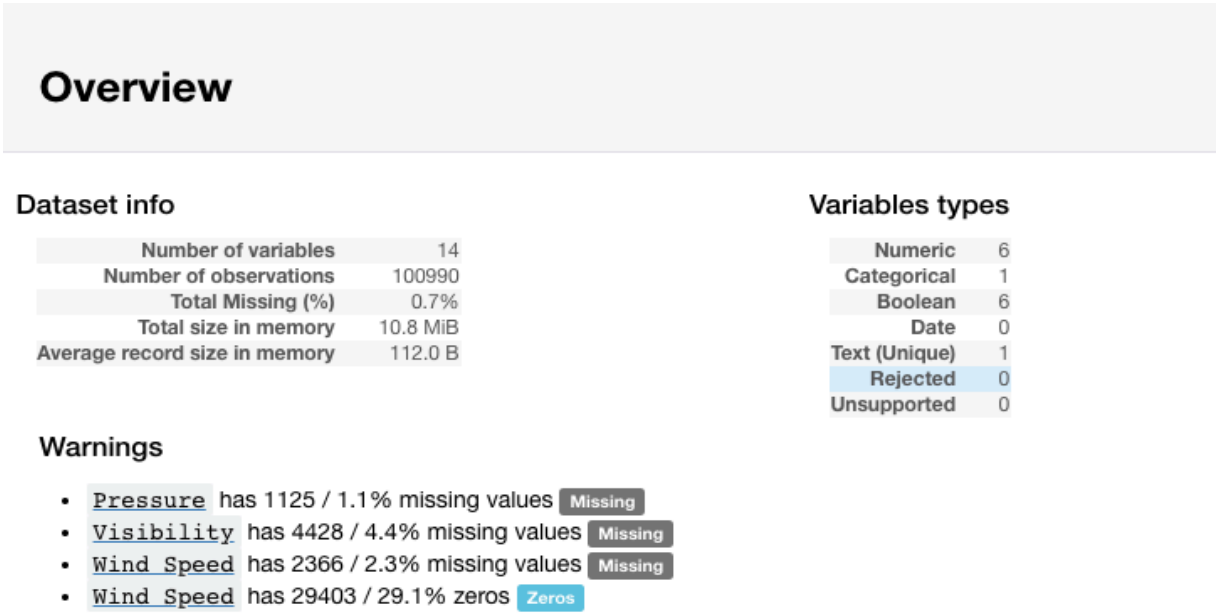
factors affecting climate change, and produce thought-provoking visualizations to increase awareness.

Data Source and Description:

This data was taken out from Kaggle which is owned by wunderground. It contains various features such as temperature, pressure, humidity, rain, etc.

Preliminary EDA

Firstly in preliminary EDA I have dropped the columns that was not complete or not useful. Some of the temperature observations were in Fahrenheit also some of Pressure, wind speed, visibility were in strange numbers so I converted them and then got the overview of each and every variables



Overview

Dataset info

Number of variables	14
Number of observations	100990
Total Missing (%)	0.7%
Total size in memory	10.8 MiB
Average record size in memory	112.0 B

Variables types

Numeric	6
Categorical	1
Boolean	6
Date	0
Text (Unique)	1
Rejected	0
Unsupported	0

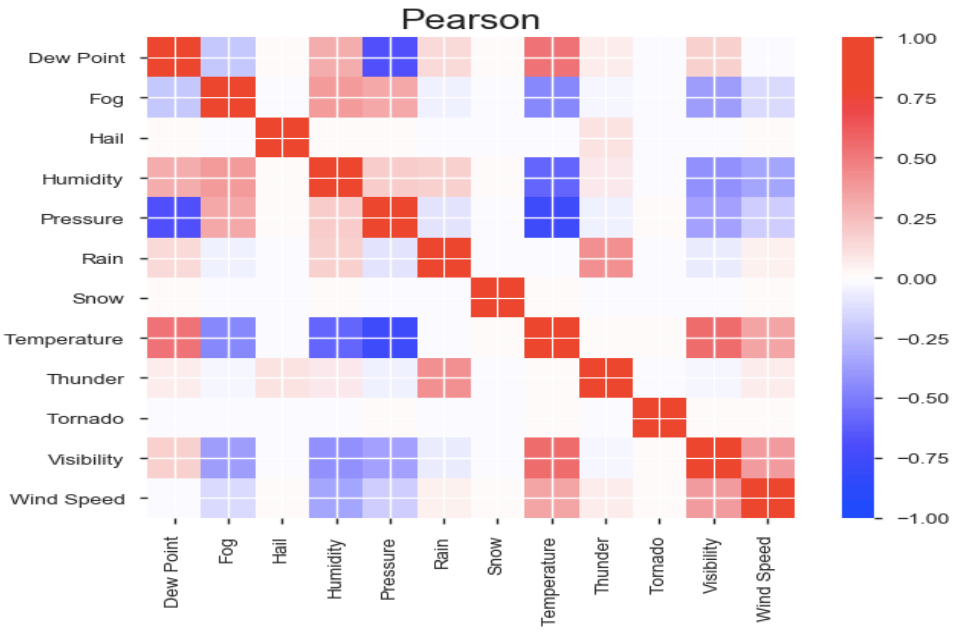
Warnings

- Pressure has 1125 / 1.1% missing values Missing
- Visibility has 4428 / 4.4% missing values Missing
- Wind Speed has 2366 / 2.3% missing values Missing
- Wind Speed has 29403 / 29.1% zeros Zeros

FINAL PROJECT STAT 517

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

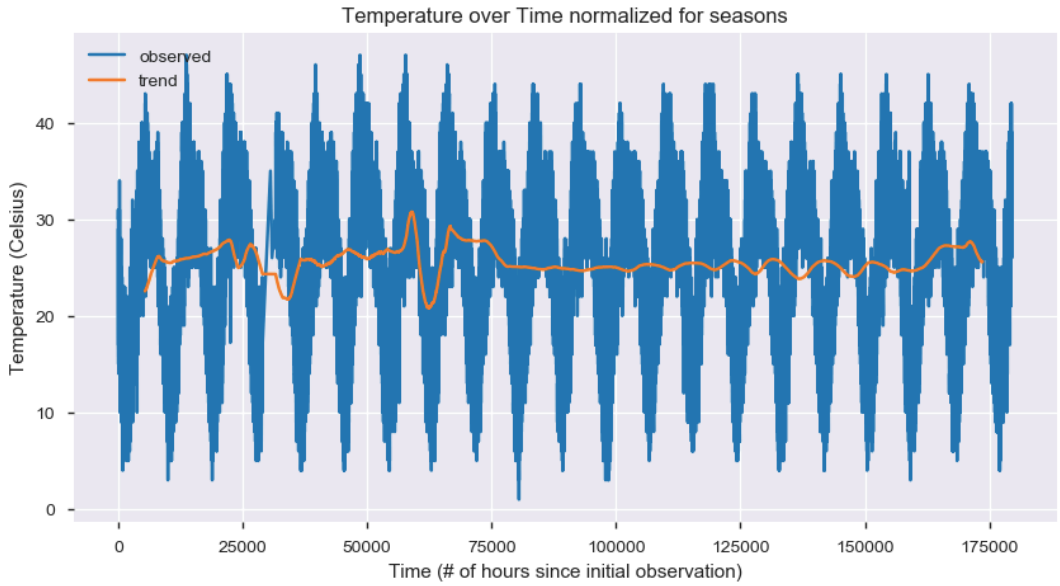
Correlations



Modelling Process

Imputation and Interpolation has been done Variable 'Condition' will require most frequent value imputation because it is categorical, and all the numerical data will require interpolation. Haze is the most common so we will impute conditions using Haze.

Visualizing trends independent of seasonality



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

The above chart shows the original data in blue, and the trend after normalizing for seasons (i.e getting rid of useless data) in orange. From this we can see that there was an abnormally cold Winter in Delhi in 2003-2004. This does not show the deadly heat wave of 2015 because it is smooth and ignores spikes.

Association Analysis

I tried using all conditions but apriori is 2^n and 51 variables takes too long so I must decrease this number. Before converting weather condition to boolean we must combine similar conditions like light haze and haze

```
data['Condition'] = data['Condition'].replace('Light Haze', 'Haze')
data['Condition'] = data['Condition'].replace('Light Fog', 'Fog')
data['Condition'] = data['Condition'].replace('Heavy Fog', 'Fog')
data['Condition'] = data['Condition'].replace('Patches of Fog', 'Fog')
data['Condition'] = data['Condition'].replace('Partial Fog', 'Fog')
data['Condition'] = data['Condition'].replace('Shallow Fog', 'Fog')
data['Condition'] = data['Condition'].replace('Light Thunderstorms and Rain', 'Thunderstorms and Rain')
data['Condition'] = data['Condition'].replace('Heavy Thunderstorms and Rain', 'Thunderstorms and Rain')
data['Condition'] = data['Condition'].replace('Light Thunderstorm', 'Thunderstorm')
data['Condition'] = data['Condition'].replace('Light Rain', 'Rain')
```

```
frequencies = apriori(booleanFeatures, min_support=0.0, use_colnames=True)
# I calculate frequencies of item sets
frequencies = frequencies[frequencies.support != 0.0]
frequencies.sort_values(by=['support'], ascending=False)
# Frequencies for single items is not useful for rule generation.
# Therefore the most frequent pairs of itemsets which occured together matters for rule generation
```

	support	itemsets
6	0.472106	(Condition_Haze)
11	0.205565	(Condition_Smoke)
7	0.092831	(Condition_Mist)
4	0.069680	(Condition_Fog)
3	0.052857	(Condition_Clear)
10	0.029122	(Condition_Rain)
13	0.028280	(Condition_Widespread Dust)
9	0.020705	(Condition_Partly Cloudy)
8	0.018447	(Condition_Overcast)
12	0.009427	(Condition_Thunderstorm)
100	0.006773	(Condition_Rain, Condition_Thunderstorm)
2	0.003743	(Condition_Blowing Sand)
5	0.000129	(Condition_Hail)
75	0.000119	(Condition_Thunderstorm, Condition_Hail)
1	0.000020	(Tornado)
23	0.000010	(Condition_Rain, Snow)
0	0.000010	(Snow)

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

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
1	(Snow)	(Condition_Rain)	0.000010	0.029122	0.000010	1.000000	34.338660	0.000010	inf
3	(Condition_Hail)	(Condition_Thunderstorm)	0.000129	0.009427	0.000119	0.923077	97.921784	0.000118	12.877453
5	(Condition_Thunderstorm)	(Condition_Rain)	0.009427	0.029122	0.006773	0.718487	24.671895	0.006498	3.448792
4	(Condition_Rain)	(Condition_Thunderstorm)	0.029122	0.009427	0.006773	0.232574	24.671895	0.006498	1.290774
2	(Condition_Thunderstorm)	(Condition_Hail)	0.009427	0.000129	0.000119	0.012605	97.921784	0.000118	1.012636
0	(Condition_Rain)	(Snow)	0.029122	0.000010	0.000010	0.000340	34.338660	0.000010	1.000330

After using apriori to get item set frequencies and association rules generation to get rules, there are six rules and some are more confidence than others. We can say with 92.3% confidence that if we observe hail then we will also observe thunder. We can say with 71.8% confidence that if we observe thunder then we will also observe rain, moreover the confidence of antecedents Snow consequents Rain is 1.00 which shows that there was only one one time snow happened in Delhi that's why the confidence is 100 % everytime. What I have done is use the apriori algorithm to find the most frequent pairs of features that occur together, then calculate the confidence P(A&&B)/P(A). High confidence means A -> B is a good rule.

Regression

I will predict temperature and visibility separately. splitted training and testing set in 90 and 10% respectively.

Linear Regression

```
lrTemp = LinearRegression().fit(xTrainTemp, yTrainTemp)
lrVis = LinearRegression().fit(xTrainVis, yTrainVis)

print("Training set score for temperature: {:.2f}".format(lrTemp.score(xTrainTemp, yTrainTemp)))
print("Test set score for temperature: {:.2f}".format(lrTemp.score(xTestTemp, yTestTemp)))
print("Training set score for visibility: {:.2f}".format(lrVis.score(xTrainVis, yTrainVis)))
print("Test set score for visibility: {:.2f}".format(lrVis.score(xTestVis, yTestVis)))
```

```
Training set score for temperature: 0.94
Test set score for temperature: 0.94
Training set score for visibility: 0.49
Test set score for visibility: 0.48
```

FINAL PROJECT STAT 517

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

Ridge Regression

```
ridgeTemp = Ridge().fit(xTrainTemp, yTrainTemp)
ridgeVis = Ridge().fit(xTrainVis, yTrainVis)

print("Training set score for temperature: {:.2f}".format(ridgeTemp.score(xTrainTemp, yTrainTemp)))
print("Test set score for temperature: {:.2f}".format(ridgeTemp.score(xTestTemp, yTestTemp)))
print("Training set score for visibility: {:.2f}".format(ridgeVis.score(xTrainVis, yTrainVis)))
print("Test set score for visibility: {:.2f}".format(ridgeVis.score(xTestVis, yTestVis)))
```

```
Training set score for temperature: 0.94
Test set score for temperature: 0.94
Training set score for visibility: 0.49
Test set score for visibility: 0.48
```

Decision Tree Regression

```
treeTemp = DecisionTreeRegressor(max_depth=5).fit(xTrainTemp, yTrainTemp)
treeVis = DecisionTreeRegressor(max_depth=5).fit(xTrainVis, yTrainVis)
print ("Tree of depth 5 Training set score for temperature: {:.2f}".format(treeTemp.score(xTrainTemp, yTrainTemp)))
print ("Tree of depth 5 Test set score for temperature: {:.2f}".format(treeTemp.score(xTestTemp, yTestTemp)))
print ("Tree of depth 5 Training set score for visibility: {:.2f}".format(treeVis.score(xTrainVis, yTrainVis)))
print ("Tree of depth 5 Test set score for visibility: {:.2f}".format(treeVis.score(xTestVis, yTestVis)))
```

```
Tree of depth 5 Training set score for temperature: 0.91
Tree of depth 5 Test set score for temperature: 0.91
Tree of depth 5 Training set score for visibility: 0.50
Tree of depth 5 Test set score for visibility: 0.49
```

Gradient Boosted Regression

```
gbrTemp = GradientBoostingRegressor()
gbrTemp.fit(xTrainTemp, yTrainTemp)
gbrVis = GradientBoostingRegressor()
gbrVis.fit(xTrainVis, yTrainVis)
print("Training set score for temperature: {:.3f}".format(gbrTemp.score(xTrainTemp, yTrainTemp)))
print("Test set score for temperature: {:.3f}".format(gbrTemp.score(xTestTemp, yTestTemp)))
print("Training set score for visibility: {:.3f}".format(gbrVis.score(xTrainVis, yTrainVis)))
print("Test set score for visibility: {:.3f}".format(gbrVis.score(xTestVis, yTestVis)))
```

```
/Users/rohit1/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: 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 ra
vel().
  y = column_or_1d(y, warn=True)
/Users/rohit1/anaconda3/lib/python3.6/site-packages/sklearn/utils/validation.py:578: 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 ra
vel().
  y = column_or_1d(y, warn=True)
```

```
Training set score for temperature: 0.978
Test set score for temperature: 0.977
Training set score for visibility: 0.572
Test set score for visibility: 0.563
```

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

K-Nearest Neighbors Regressor

At 7 neighbors the model predicted visibility the best, but still much worse than temperature. All models could not predict visibility as well as temperature which suggests that visibility is a hard feature to predict.

```
Training set score with 4 neighbors for temperature: 0.99
Test set score with 4 neighbors for temperature: 0.97
Training set score with 4 neighbors for visibility: 0.72
Test set score with 4 neighbors for visibility: 0.52
Training set score with 5 neighbors for temperature: 0.98
Test set score with 5 neighbors for temperature: 0.97
Training set score with 5 neighbors for visibility: 0.69
Test set score with 5 neighbors for visibility: 0.53
Training set score with 6 neighbors for temperature: 0.98
Test set score with 6 neighbors for temperature: 0.97
Training set score with 6 neighbors for visibility: 0.67
Test set score with 6 neighbors for visibility: 0.53
Training set score with 7 neighbors for temperature: 0.98
Test set score with 7 neighbors for temperature: 0.98
Training set score with 7 neighbors for visibility: 0.66
Test set score with 7 neighbors for visibility: 0.54
Training set score with 8 neighbors for temperature: 0.98
Test set score with 8 neighbors for temperature: 0.98
Training set score with 8 neighbors for visibility: 0.65
Test set score with 8 neighbors for visibility: 0.54
```

Clustering

The first problem I had with clustering was the computer would crash because the dataset was too big therefore I have taken one observation per day instead of hourly.

KMeans

```
kmeans = KMeans(n_clusters=20, random_state=int(time.time()))
kmeans.fit(clusterData)
kmeansLabels = kmeans.predict(clusterData)
```

GMM

```
gmm = GaussianMixture(n_components=20, covariance_type='full').fit(clusterData)
gmmLabels = gmm.predict(clusterData)
```

Agglomerative Clusutering

```
agglo = AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto', connectivity=None, linkage='ward',
agglo.fit(clusterData)
aggloLabels = agglo.labels_
```

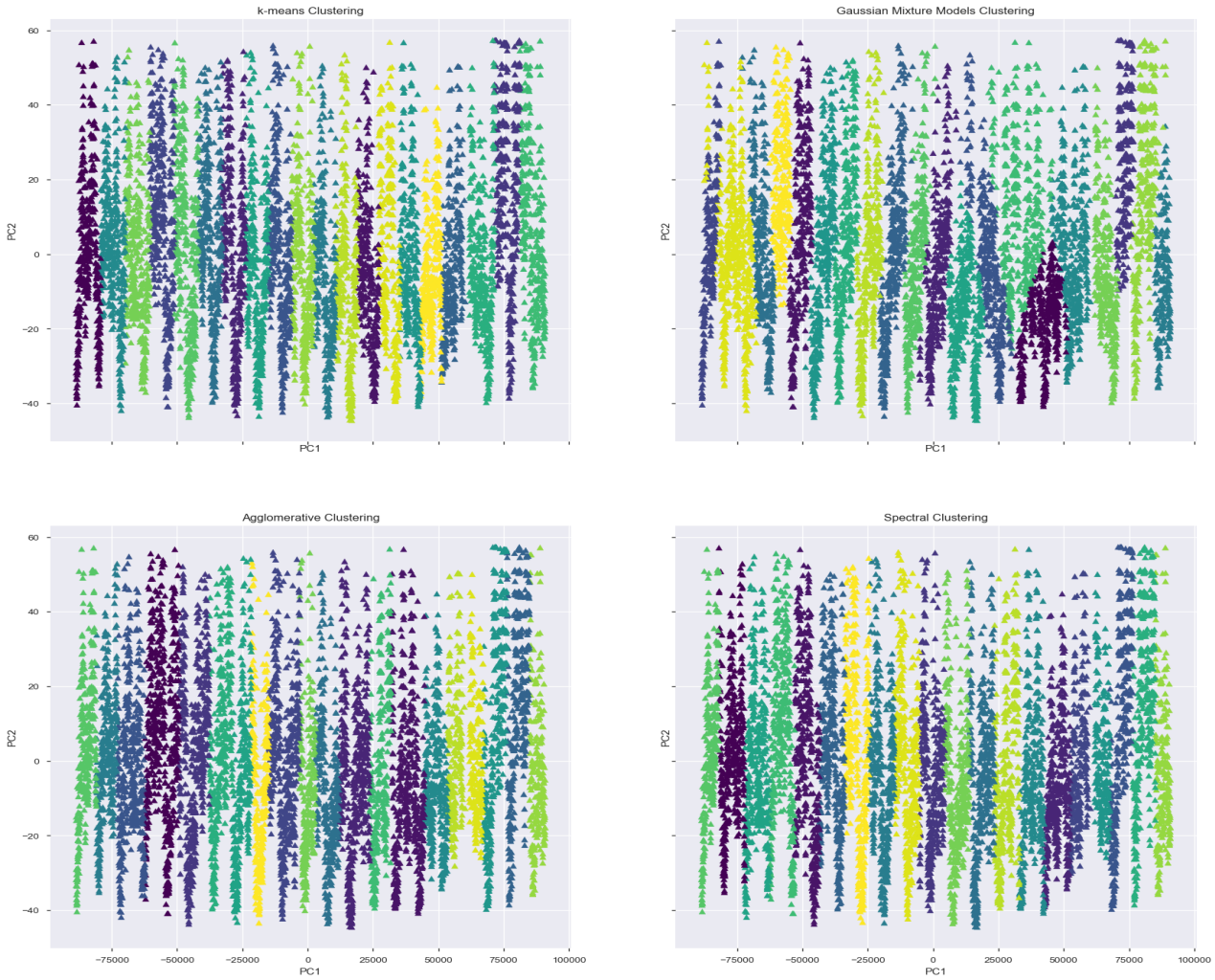
Spectral Clustering

```
spectral = SpectralClustering(n_clusters=20, affinity='nearest_neighbors', assign_labels='kmeans')
spectral.fit(clusterData)
spectralLabels = spectral.labels_
```

```
/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"
```

FINAL PROJECT STAT 517

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



I have used KMeans, GMM, Agglomerative and spectral clustering for visualization. Even after PCA dimensionality reduction, the data is clustered into 20 stripes probably the 20 years. This means that PCA found the most important component to be time, and reduced all other features into PC2. The different colors represents the different weather conditions.

Categorical Classification

I try to classify the observations into weather conditions.

```
x = data[['Dew Point','Humidity','Pressure','Temperature','Visibility','Wind Speed']]
conditions = dict(enumerate(list(data['Condition'].value_counts().keys())))
transfer = {conditions[n]:n for n in range(len(conditions))}
y = np.array([transfer[n] for n in list(data['Condition'])])
```

FINAL PROJECT STAT 517

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

Categorical Classification without PCA

```
X = data[['Dew Point','Humidity','Pressure','Temperature','Visibility','Wind Speed']]
X = np.array(X)
X = StandardScaler().fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=int(time.time()))

names = [
    "Nearest Neighbors",
    "Decision Tree",
    "Random Forest",
    "Neural Net",
#    "AdaBoost",
    "Naive Bayes"]
#    "QDA"]

classifiers = [
    KNeighborsClassifier(3),
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    MLPClassifier(alpha=1),
#    AdaBoostClassifier(),
    GaussianNB()]
#    QuadraticDiscriminantAnalysis()]

# iterate over classifiers
for name, clf in zip(names, classifiers):
    clf.fit(X_train, y_train)
    print(name + ' Training set score: {:.2f}%'.format(clf.score(X_train, y_train)*100))
    print(name + ' Testing set score: {:.2f}%'.format(clf.score(X_test, y_test)*100))
```

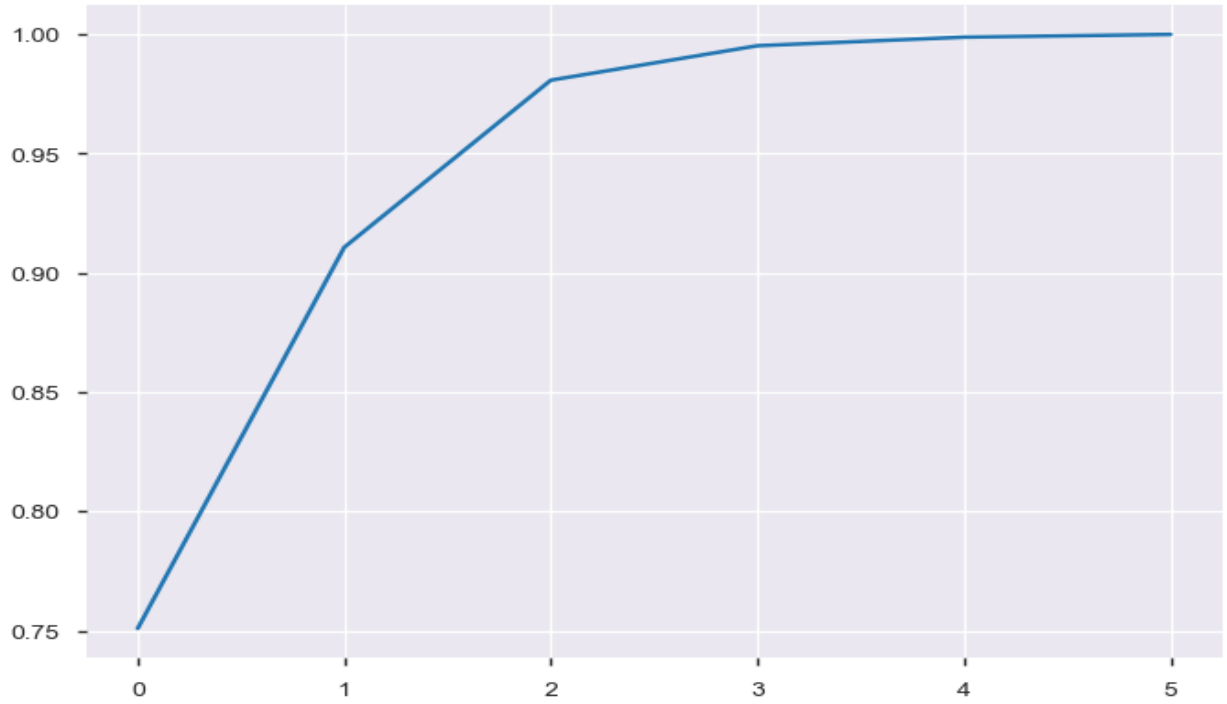
```
Nearest Neighbors Training set score: 82.83%
Nearest Neighbors Testing set score: 72.09%
Decision Tree Training set score: 70.72%
Decision Tree Testing set score: 71.30%
Random Forest Training set score: 67.92%
Random Forest Testing set score: 67.68%
Neural Net Training set score: 70.41%
Neural Net Testing set score: 70.34%
Naive Bayes Training set score: 62.79%
Naive Bayes Testing set score: 62.75%
```

Categorical Classification with PCA

```
pca = PCA().fit(x)
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()
```

FINAL PROJECT STAT 517

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



The number of principle components with the highest ratio of variance to components is 1

Using 1 components will preserve 91.08% of the data

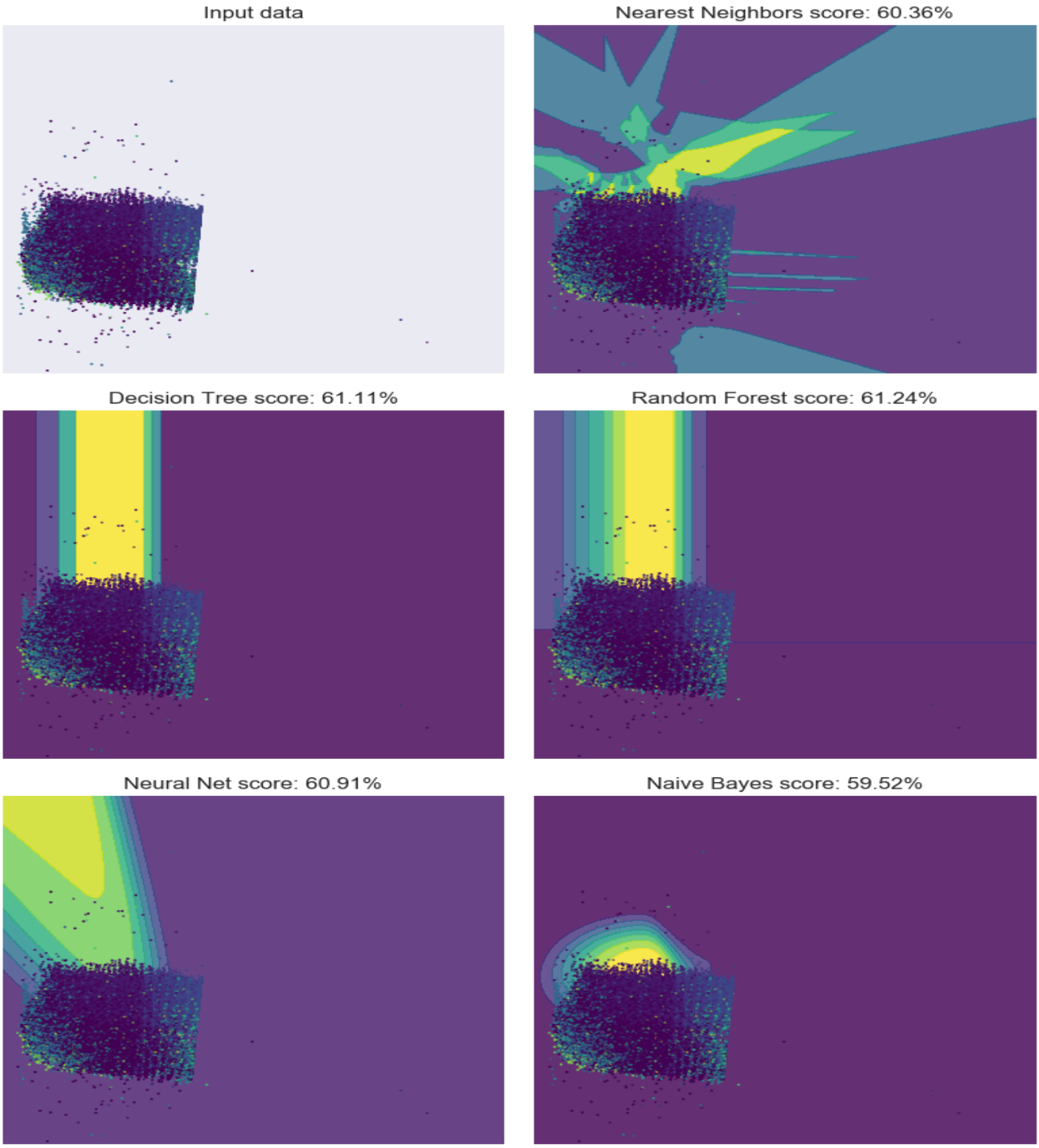
Even though the highest ratio is 1 component, this is not enough so I will use 2.

```
pca = PCA(n_components=2)
pca.fit(x)
X = pca.transform(x)
X = StandardScaler().fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=int(time.time()))
```

Next I test all the algorithms on this problem which are Nearest Neighbors, Decision Tree, Random Forest, Neural Net, Naive Bayes.

FINAL PROJECT STAT 517

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



This classification problem is very hard for these algorithms. KNN finds interesting patterns in the data but none of the scores are good. As you can see that the different weather conditions are mixed together and separating them is almost impossible for these algorithms to the regions at the background which are basically different weather conditions in different colors. One reason

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

also why it's difficult in separation because of all the dataset couldn't recognize properly the different patterns colors of matching weather condition color in the background. It is interesting that without PCA reduction these algorithms performed more accurately, but are harder to visualize.

Binary Classification

I classify the observations into hot or cold (above average or below average).

```
x = pd.concat([pd.DataFrame(data.drop(['Date','Temperature','Condition','Fog','Hail','Rain','Thunder'],axis=1)), ca
avgTemp = float(data['Temperature'].sum())/len(data['Temperature'])
# I convert temperature to binary hot or cold by calculating the average temperature
# and dividing the dataset into hotter than average and colder than average.
y = np.array([1 if n < avgTemp else 0 for n in list(data['Temperature'])])
```

Binary Classification without PCA

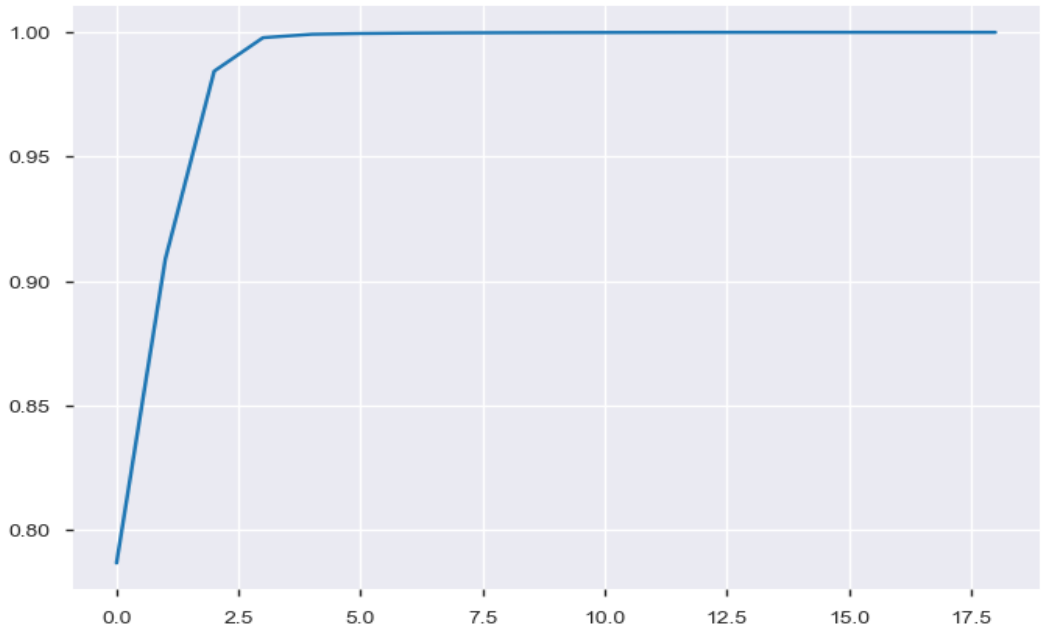
```
X = StandardScaler().fit_transform(x)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=int(time.time()))

names = [
    "Nearest Neighbors",
    "Decision Tree",
    "Random Forest",
    "Neural Net",
#    "AdaBoost",
    "Naive Bayes"]
#    "QDA"]

classifiers = [
    KNeighborsClassifier(3),
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    MLPClassifier(alpha=1),
#    AdaBoostClassifier(),
    GaussianNB()]
#    QuadraticDiscriminantAnalysis()]

# iterate over classifiers
for name, clf in zip(names, classifiers):
    clf.fit(X_train, y_train)
    print(name + ' Training set score: {:.2f}%'.format(clf.score(X_train, y_train)*100))
    print(name + ' Testing set score: {:.2f}%'.format(clf.score(X_test, y_test)*100))
```

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

```
Nearest Neighbors Training set score: 98.14%
Nearest Neighbors Testing set score: 96.43%
Decision Tree Training set score: 93.93%
Decision Tree Testing set score: 93.75%
Random Forest Training set score: 90.25%
Random Forest Testing set score: 90.13%
Neural Net Training set score: 96.37%
Neural Net Testing set score: 96.46%
Naive Bayes Training set score: 47.98%
Naive Bayes Testing set score: 47.48%
```

Binary Classification with PCA

```
pca = PCA().fit(x)
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()
```



The number of principle components with the highest ratio of variance to components is 2

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

Using 2 components will preserve 98.44% of the data.

I will use 2 components

```
pca = PCA(n_components=2)
pca.fit(x)
X = pca.transform(x)
X = StandardScaler().fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=int(time.time()))
```

Next I will use code from sklearn documentation to test all classification algorithms on this problem.

```
fig, ax = plt.subplots(3, 2, sharex='col', sharey='row', figsize=(8, 12))

cmp = plt.cm.RdBu
cmp_bright = ListedColormap(['#FF0000', '#0000FF'])

names = [
    "Nearest Neighbors",
    "Decision Tree",
    "Random Forest",
    "Neural Net",
#    "AdaBoost",
    "Naive Bayes"]
#    "QDA"]

classifiers = [
    KNeighborsClassifier(3),
    DecisionTreeClassifier(max_depth=5),
    RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1),
    MLPClassifier(alpha=1),
#    AdaBoostClassifier(),
    GaussianNB()]
#    QuadraticDiscriminantAnalysis()]

h = .02  # step size in the mesh

rng = np.random.RandomState(2)
X += 2 * rng.uniform(size=X.shape)

i = 0

x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))
```

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

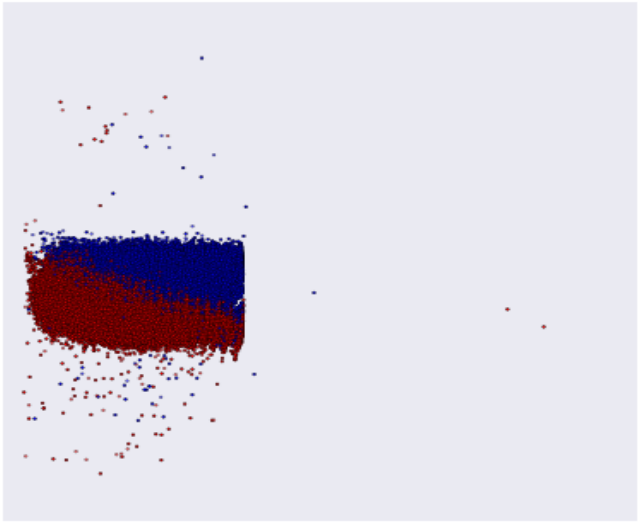
```
# just plot the dataset first
ax[int(i/2),i%2].set_title("Input data")
# Plot the training points
ax[int(i/2),i%2].scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cmp_bright, edgecolors='k', s=1.5)
# Plot the testing points
ax[int(i/2),i%2].scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cmp_bright, alpha=0.6, edgecolors='k', s=1.5)
ax[int(i/2),i%2].set_xlim(xx.min(), xx.max())
ax[int(i/2),i%2].set_ylim(yy.min(), yy.max())
ax[int(i/2),i%2].set_xticks(())
ax[int(i/2),i%2].set_yticks(())
i += 1

scoresPCA = {}
# iterate over classifiers
scores = {}
for name, clf in zip(names, classifiers):
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    # Plot the decision boundary. For that, we will assign a color to each
    # point in the mesh [x_min, x_max]x[y_min, y_max].
    if hasattr(clf, "decision_function"):
        Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
        scores[name] = clf.decision_function(X_test)
    else:
        Z = clf.predict_proba(np.c_[xx.ravel(), yy.ravel()])[:, 1]
        scores[name] = clf.predict_proba(X_test)

    # Put the result into a color plot
    Z = Z.reshape(xx.shape)
    ax[int(i/2),i%2].contourf(xx, yy, Z, cmap=cmp, alpha=.8)
```

```
    # Plot the training points
    ax[int(i/2),i%2].scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cmp_bright, edgecolors='k', s=1.5)
    # Plot the testing points
    ax[int(i/2),i%2].scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cmp_bright, edgecolors='k', alpha=0.6, s=1.

    ax[int(i/2),i%2].set_xlim(xx.min(), xx.max())
    ax[int(i/2),i%2].set_ylim(yy.min(), yy.max())
    ax[int(i/2),i%2].set_xticks(())
    ax[int(i/2),i%2].set_yticks(())
    ax[int(i/2),i%2].set_title(name + ' score: {:.2f}%'.format(score*100))
    i += 1

plt.tight_layout()
plt.show()
```

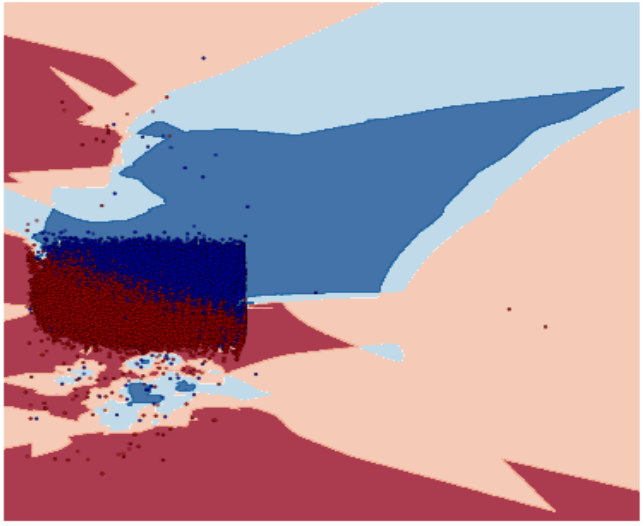
FINAL PROJECT STAT 517

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

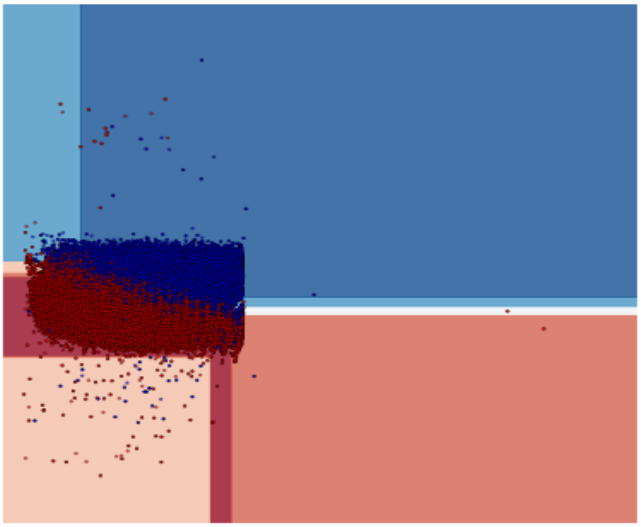
Input data



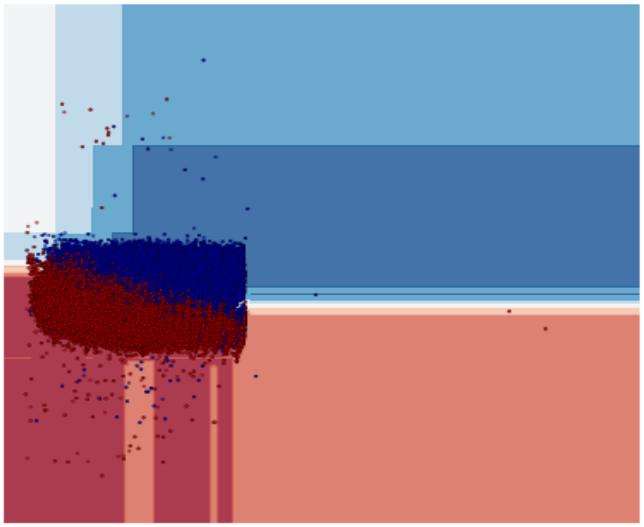
Nearest Neighbors score: 92.56%



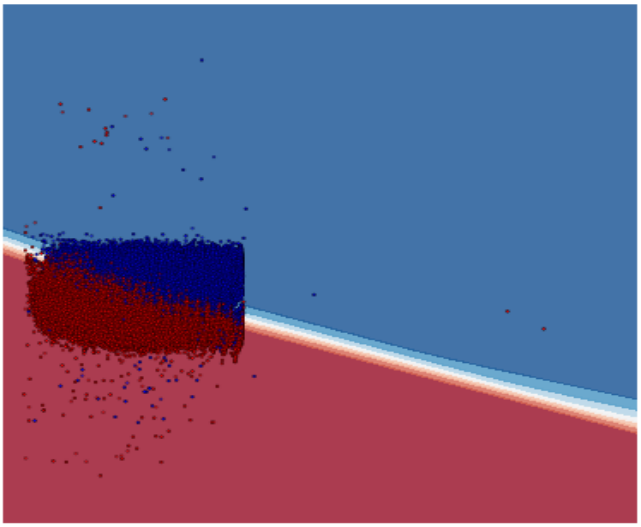
Decision Tree score: 92.26%



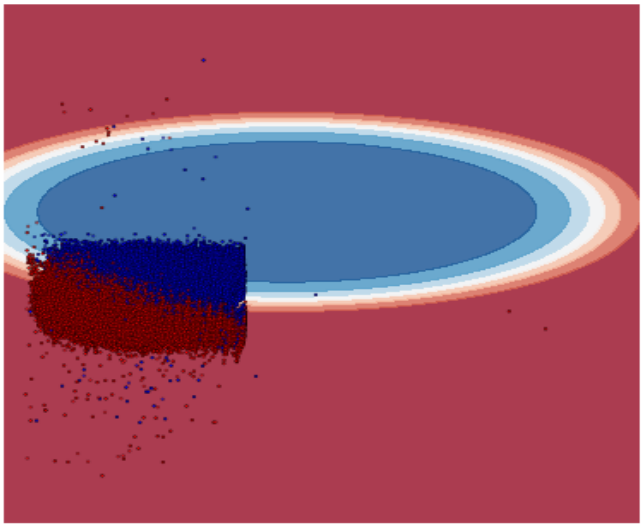
Random Forest score: 92.38%



Neural Net score: 92.46%



Naive Bayes score: 90.76%



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

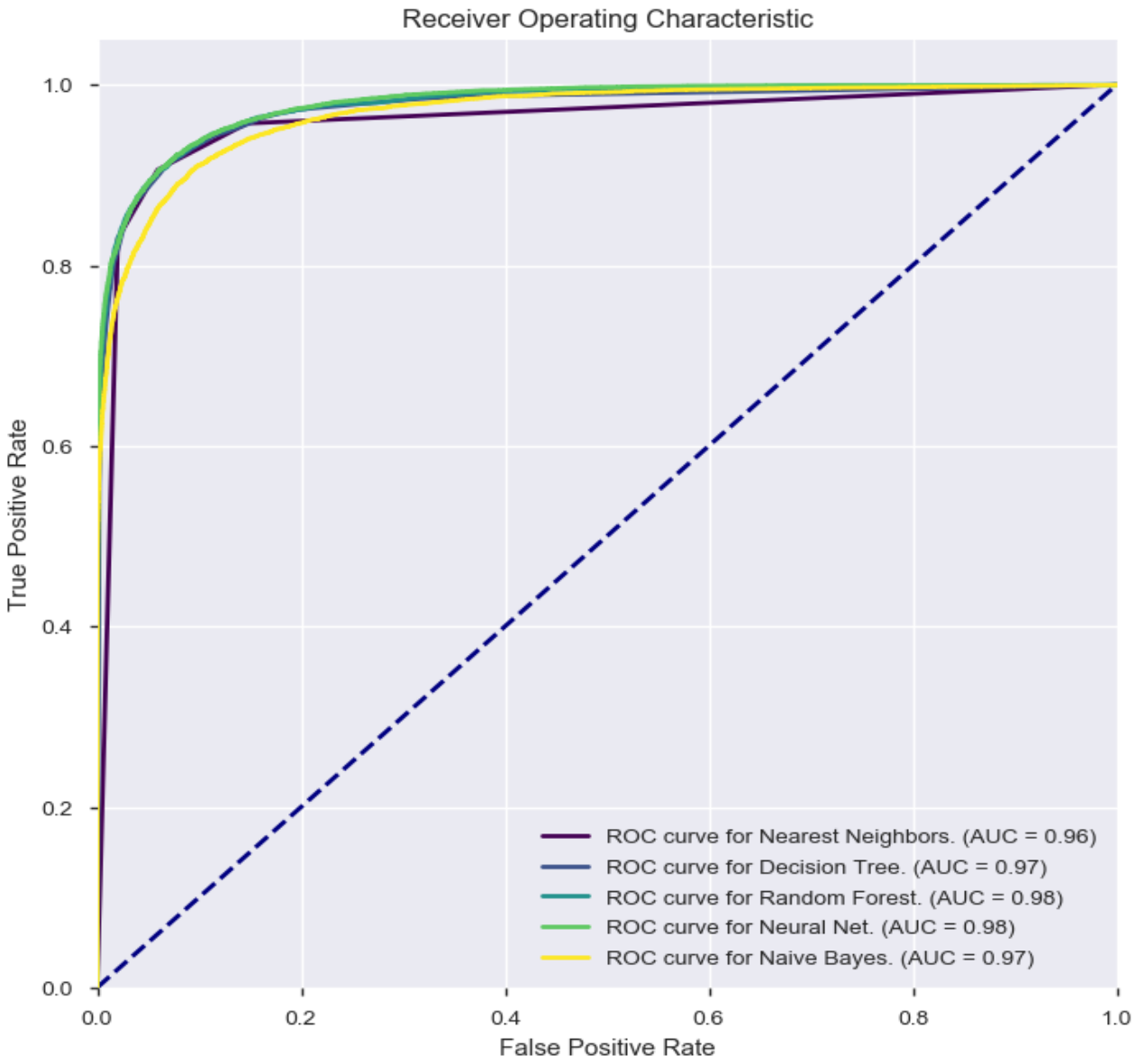
In the above graph the regions underneath the data points are showing where the model would classify points if they existed and all the different colors represent different weather conditions. These models can classify points in the entire space even though these points are mostly on the left side. I have shown that these models can classify the dataset into hot and cold (where hot is red and cold is blue regions).

The hot/cold classification problem is much more interesting and it's more easy for these algorithms. They all found good solutions and KNN even found some of the hot points on the bottom. ROC will tell if these are accurate classifiers or not.

ROC Visualization of Binary Classification

```
viridis = plt.cm.get_cmap('viridis', len(names))
plt.figure(figsize=(8,8))
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')

for n, name in enumerate(names):
    fpr, tpr, thresholds = roc_curve(y_test, scores[name][:, 1])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, color=viridis(n), lw=2, label='ROC curve for ' + name + '. (AUC = %0.2f)' % roc_auc)
plt.legend(loc="lower right")
plt.show()
```

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



The area under the curve (lower right legend) is equal to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. We want the AUC to be as high as possible because that means the curve is as far to the top left as possible and at the elbow there is a low false positive rate and a high true positive rate.

Conclusion and Future Work

By all the analysis and visualization we can predict the weather condition in future as In our regression model temperature was the easiest to predict and hd higher accuracy, however all

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

models could not predict visibility which suggests that visibility is a hard feature to predict also It was harder for all the algorithms in categorical classification to classify and separate all the mixed weather conditions. In Binary hot/cold classification problem was much more interesting and it's easier for these algorithms to find the good solutions and ROC graph for all the algorithms showed it the good classifier. My future work would be testing the multi-label classification which I tried testing, but my system crashed and also will try to test with all the conditions without merging similar condition into once.

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

References

1. UNHCR - The UN Refugee Agency. http://www.unhcr.org/en-us/news/latest/2008/12/493e9bd94/top-unhcr-official-warns-displacement-climate-change.html. [Ac- cessed: 2017-04-10].

2. Fast Facts About Climate Change - National Wildlife Federation. https://www.nwf.org/Eco-Schools-USA/Become-an-Eco-School/Pathways/Climate-Change/Facts.aspx. [Accessed: 2017-04-04].

3. The Guardian - 'A tipping point': record number of Americans see global warming as threat. hhttps://www.theguardian.com/environment/2016/mar/18/ climate-change-record-concern-us-global-warming-poll. [Accessed: 2015-04-10].