In [10]:

```
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).
```

In [11]:

```
import sqlite3
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from IPython.display import display
from sklearn import tree
from sklearn.manifold import TSNE
from sklearn import svm
from sklearn.svm import SVC
from sklearn import linear_model
from sklearn.externals import joblib
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import log_loss
from sklearn.metrics import confusion_matrix
from sklearn.multiclass import OneVsRestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import RandomizedSearchCV
```

In [12]:

```
#Original Dataset
conn = sqlite3.connect('drive/My Drive/CaseStudy1/FPA_FOD_20170508.sqlite')
```

In [13]:

```
df = pd.read_sql_query("SELECT * FROM fires;", conn)
```

In [14]:

```
#Taking random 4500 rows of the entire datset to get their fire size class prediction, We
can take any number of rows from any part of the data for getting the result
dataset = df[5000:50000:10]
dataset_func1 = dataset.drop(['FIRE_SIZE_CLASS'], axis = 1)
print(dataset_func1)
```

```
       OBJECTID  ...                                              Shape
5000       5001  ...  b'\x00\x01\xad\x10\x00\x00\xd0\xb7\x92>\xe9\x0...
5010       5011  ...  b'\x00\x01\xad\x10\x00\x00\xa0p=\n\xd7\x0b\\\x...
5020       5021  ...  b'\x00\x01\xad\x10\x00\x00\xfc\xff\xff\xff\xff...
5030       5031  ...  b'\x00\x01\xad\x10\x00\x00\xa4]3\x96\xfc\x02^\...
5040       5041  ...  b'\x00\x01\xad\x10\x00\x00D\xe1z\x14\xae\xef[\...
...         ...  ...                                                ...
49950     49951  ...  b'\x00\x01\xad\x10\x00\x00$o\x996\xd0\x19^\xc0...
49960     49961  ...  b'\x00\x01\xad\x10\x00\x00\x14\xaeG\xe1z,^\xc0...
49970     49971  ...  b'\x00\x01\xad\x10\x00\x00\xd4\xa3p=\n/^\xc0\x...
49980     49981  ...  b'\x00\x01\xad\x10\x00\x00\xf4\x15R\x1b\xe8L^\...
49990     49991  ...  b'\x00\x01\xad\x10\x00\x00\xf4\x15R\x1b\xe8L^\...

[4500 rows x 38 columns]
```

In [15]:

```
def DataPrediction(data):
  test_df = pd.DataFrame()
  for i in range(3):
    SampleModel = joblib.load('drive/My Drive/CaseStudy1/SampleModel_'+ str(i) + '.pkl')
    predictedValues = SampleModel.predict(data)
    columnName = 'predict' + str(i)
    test_df[columnName] = predictedValues

  test_finalPrediction = []
  for j in range(len(test_df)):
    row_list = test_df.iloc[j].values.tolist()
    majority_count = max(set(row_list) , key=row_list.count)
    test_finalPrediction.append(majority_count)

  test_finalPrediction = np.array(test_finalPrediction)
  return(test_finalPrediction)
```

In [16]:

```
def function1(data):
  '''This function will give the prediction for input data given'''

  print('Deleting unnecessary features......\n')
  del_features = ['OBJECTID', 'FOD_ID', 'FPA_ID', 'NWCG_REPORTING_UNIT_ID', 'NWCG_REPORTING_UNIT_NAME', 'SOURCE_REPORTING_UNIT', 'SOURCE_REPORTING_UNIT_NAME', 'LOCAL_FIRE_REPORT_ID', 'LOCAL_INCIDENT_ID', 'FIRE_CODE', 'FIRE_NAME', 'ICS_209_INCIDENT_NUMBER' , 'ICS_209_NAME', 'MTBS_FIRE_NAME', 'MTBS_ID', 'COMPLEX_NAME', 'DISCOVERY_DATE', 'STAT_CAUSE_DESCR', 'CONT_DATE', 'CONT_TIME', 'FIRE_SIZE', 'OWNER_DESCR', 'COUNTY', 'FIPS_CODE', 'FIPS_NAME' , 'Shape' ]
  for i, item in enumerate(del_features):
    del data[item];
  print('Data shape is: ', data.shape, '\n')

  print('Encoding features......\n')
  label_encoder = preprocessing.LabelEncoder()
  encode_features = ['SOURCE_SYSTEM_TYPE', 'SOURCE_SYSTEM', 'NWCG_REPORTING_AGENCY']
  for j, e_item in enumerate(encode_features):
    data[e_item] = label_encoder.fit_transform(data[e_item])
    data[e_item].astype('int64')

  #Manually encoding states feature
  data['STATE'] = data['STATE'].map({'AL': 0, 'AK': 1, 'AZ': 2, 'AR': 3, 'CA': 4, 'CO': 5,'CT': 6,'DE': 7,'DC': 8,'FL': 9,'GA': 10,'HI': 11,'ID': 12,'IL': 13,'IN': 14,'IA': 15,'KS': 16,'KY': 17,'LA': 18,'ME': 19,'MD': 20,'MA': 21,'MI': 22,'MN': 23,'MS': 24,'MO': 25,'MT': 26,'NE': 27,'NV': 28,'NH': 29,'NJ': 30,'NM': 31,'NY': 32,'NC': 33,'ND': 34,'OH': 35,'OK': 36,'OR': 37,'PA': 38,'PR': 39,'RI': 40,'SC': 41,'SD': 42,'TN': 43,'TX': 44,'UT': 45,'VT': 46,'VA': 47,'WA': 48,'WV': 49,'WI': 50,'WY': 51})
  data['STATE'].astype('int64')

  print('Performing Feature Engineering......\n')
  #Adding Feature Discovery Month
  discovery_month = [];
  for i in range(len(data)):
   key = data.iloc[i]['DISCOVERY_DOY']
   if( 1 <= key <= 31 ):
    discovery_month.append(1)
   elif ( 32 <= key <= 60 ):
      discovery_month.append(2)
   elif ( 61 <= key <= 91 ):
     discovery_month.append(3)
   elif ( 92 <= key <= 121 ):
     discovery_month.append(4)
   elif ( 122 <= key <= 152 ):
     discovery_month.append(5)
   elif ( 153 <= key <= 182 ):
     discovery_month.append(6)
   elif ( 183 <= key <= 213 ):
     discovery_month.append(7)
   elif ( 214 <= key <= 244 ):
     discovery_month.append(8)
   elif ( 245 <= key <= 274 ):
     discovery_month.append(9)
```

```
 elif ( 275 <= key <= 305 ):
   discovery_month.append(10)
 elif ( 306 <= key <= 335 ):
   discovery_month.append(11)
 elif ( 336 <= key <= 366 ):
   discovery_month.append(12)

data['DISCOVERY_MONTH'] = discovery_month
data['DISCOVERY_MONTH'].astype('int64')
print('Data shape is: ', data.shape, '\n')

#Delete DISCOVERY_DOY and CONT_DOY also now
del data['DISCOVERY_DOY']
del data['CONT_DOY']

#Feature2 DISCOVERY_TOD
discovery_tod = [];
data['DISCOVERY_TIME'] = data['DISCOVERY_TIME'].replace([None],'0000')
for i in range(len(data)):
  key = data.iloc[i]['DISCOVERY_TIME']
  if( key == '0000' ):
    discovery_tod.append(0)
  elif ( '0000' < key <= '0600' ):
    discovery_tod.append(1)
  elif ( '0600' < key <= '1200' ):
    discovery_tod.append(2)
  elif ( '1200' < key <= '1600' ):
    discovery_tod.append(3)
  elif ( '1600' < key <= '2000' ):
    discovery_tod.append(4)
  elif ( '2000' < key <= '2400' ):
    discovery_tod.append(5)

data['DISCOVERY_TOD'] = discovery_tod
data['DISCOVERY_TOD'].astype('int64')

del data['DISCOVERY_TIME']

data['LATITUDE'] = (data['LATITUDE']*10).apply(np.floor)/10
data['LONGITUDE'] = (data['LONGITUDE']*10).apply(np.floor)/10

#Add forest Area feature
forest_Area = pd.read_excel('drive/My Drive/CaseStudy1/FOREST_Area.xlsx')
forest_Area.head()
STATE_PRCNT_FOREST = [];
for i in range(len(data)):
  key = data.iloc[i]['STATE'].astype('int64')
  STATE_PRCNT_FOREST.append(forest_Area['Forest_Coverage'].values[key])

data['STATE_PRCNT_FOREST'] = STATE_PRCNT_FOREST
data['STATE_PRCNT_FOREST'].astype('float64')

#Add Avg Temp Feature
avg_temp  = pd.read_excel('drive/My Drive/CaseStudy1/avg_temp.xlsx')

AVG_TEMP_LIST = [];
for i in range(len(data)):
  state_key = data.iloc[i]['STATE'].astype('int64')
  year_key = data.iloc[i]['FIRE_YEAR'].astype('int64')
  AVG_TEMP_LIST.append(avg_temp[year_key].values[state_key])

data['AVG_TEMP'] = AVG_TEMP_LIST
data['AVG_TEMP'].astype('float64')

#Add Avg Prec Feature
avg_prec  = pd.read_excel('drive/My Drive/CaseStudy1/avg_prec.xlsx')

AVG_PREC_LIST = [];
for i in range(len(data)):
 state_key = data.iloc[i]['STATE'].astype('int64')
 AVG_PREC_LIST.append(avg_prec['Avg_Prec'].values[state_key])
```

```
  data['AVG_PREC'] = AVG_PREC_LIST
  data['AVG_PREC'].astype('float64')
  print('Final features are: ', data.columns,'\n')
  print('EDA Completed...... \n')
  print('Predicting the fire size class......\n')

  predictions = DataPrediction(data)
  data['PREDICTED_CLASS'] = predictions
  #Simplifying the predicted class by giving area covered in each class
  predictedRange = []
  for i in range(len(data)):
    key = data.iloc[i]['PREDICTED_CLASS']
    if( key == 1 ):
      predictedRange.append('0-0.25 acres')
    elif ( key == 2 ):
      predictedRange.append('0.26-9.9 acres')
    elif ( key == 3 ):
      predictedRange.append('10.0-99.9 acres')
    elif ( key == 4 ):
      predictedRange.append('100-299 acres')
    elif ( key == 5 ):
      predictedRange.append('300-999 acres')
    elif ( key == 6 ):
      predictedRange.append('1000-5000 acres')
    else:
      predictedRange.append('5000+ acres')

  data['Area Range'] = predictedRange

  print(data)
```

In [17]:

```
function1(dataset_func1)
```

```
Deleting unnecessary features......

Data shape is:  (4500, 12)

Encoding features......

Performing Feature Engineering......

Data shape is:  (4500, 13)

Final features are:  Index(['SOURCE_SYSTEM_TYPE', 'SOURCE_SYSTEM', 'NWCG_REPORTING_AGENCY
',
       'FIRE_YEAR', 'STAT_CAUSE_CODE', 'LATITUDE', 'LONGITUDE', 'OWNER_CODE',
       'STATE', 'DISCOVERY_MONTH', 'DISCOVERY_TOD', 'STATE_PRCNT_FOREST',
       'AVG_TEMP', 'AVG_PREC'],
      dtype='object')

EDA Completed......

Predicting the fire size class......

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  13 out of  13 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  14 out of  14 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  15 out of  15 | elapsed:    0.0s remaining:    0.0s
```

```
[Parallel(n_jobs=1)]: Done  16 out of  16 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  17 out of  17 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  18 out of  18 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  19 out of  19 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  21 out of  21 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  22 out of  22 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  23 out of  23 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  24 out of  24 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  26 out of  26 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  28 out of  28 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  29 out of  29 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  31 out of  31 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  32 out of  32 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  33 out of  33 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  34 out of  34 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  35 out of  35 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  36 out of  36 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  37 out of  37 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  38 out of  38 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  39 out of  39 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  41 out of  41 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  42 out of  42 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  43 out of  43 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  44 out of  44 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  45 out of  45 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  46 out of  46 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  47 out of  47 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  48 out of  48 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  49 out of  49 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 110 out of 110 | elapsed:    0.1s finished
       SOURCE_SYSTEM_TYPE  SOURCE_SYSTEM  ...  PREDICTED_CLASS      Area Range
5000                    0              0  ...                1    0-0.25 acres
5010                    0              0  ...                1    0-0.25 acres
5020                    0              0  ...                2  0.26-9.9 acres
5030                    0              0  ...                1    0-0.25 acres
5040                    0              0  ...                1    0-0.25 acres
...                   ...            ...  ...              ...             ...
49950                   0              0  ...                1    0-0.25 acres
49960                   0              0  ...                1    0-0.25 acres
49970                   0              0  ...                1    0-0.25 acres
49980                   0              0  ...                1    0-0.25 acres
49990                   0              0  ...                1    0-0.25 acres

[4500 rows x 16 columns]
```

Result: 2 columns namely 'PREDICTED_CLASS' and 'Area Range' are the results obtained for the given set of features in our model. We can change the number of input features as per our convinience.

In [18]:

```
#Function 2: Taking both x and y values to get the the Performance Metric values for our given data
def function2(dataset):
  '''Taking the entire dataset as input for this function and then using the labels to determine MAE and MAPE scores'''
  #Encoding y_data for computation purpose
  dataset['FIRE_SIZE_CLASS'] = dataset['FIRE_SIZE_CLASS'].map({'A': 1, 'B': 2, 'C':3, 'D':4, 'E': 5, 'F': 6,'G': 7})
  dataset['FIRE_SIZE_CLASS'].astype('int64')

  #Breaking down  features and label data

  y_data = dataset['FIRE_SIZE_CLASS']
  x_data = dataset.drop(['FIRE_SIZE_CLASS'], axis = 1)
```

```python
  #Predicing labels using x_data (Similar to function 1)
  del_features = ['OBJECTID', 'FOD_ID', 'FPA_ID', 'NWCG_REPORTING_UNIT_ID', 'NWCG_REPORTING_UNIT_NAME', 'SOURCE_REPORTING_UNIT', 'SOURCE_REPORTING_UNIT_NAME', 'LOCAL_FIRE_REPORT_ID', 'LOCAL_INCIDENT_ID', 'FIRE_CODE', 'FIRE_NAME', 'ICS_209_INCIDENT_NUMBER' , 'ICS_209_NAME', 'MTBS_FIRE_NAME', 'MTBS_ID', 'COMPLEX_NAME', 'DISCOVERY_DATE', 'STAT_CAUSE_DESCR', 'CONT_DATE', 'CONT_TIME', 'FIRE_SIZE', 'OWNER_DESCR', 'COUNTY', 'FIPS_CODE', 'FIPS_NAME' , 'Shape' ]
  for i, item in enumerate(del_features):
    del x_data[item];


  label_encoder = preprocessing.LabelEncoder()
  encode_features = ['SOURCE_SYSTEM_TYPE', 'SOURCE_SYSTEM', 'NWCG_REPORTING_AGENCY']
  for j, e_item in enumerate(encode_features):
    x_data[e_item] = label_encoder.fit_transform(x_data[e_item])
    x_data[e_item].astype('int64')

  #Manually encoding states feature
  x_data['STATE'] = x_data['STATE'].map({'AL': 0, 'AK': 1, 'AZ': 2, 'AR': 3, 'CA': 4, 'CO': 5,'CT': 6,'DE': 7,'DC': 8,'FL': 9,'GA': 10,'HI': 11,'ID': 12,'IL': 13,'IN': 14,'IA': 15,'KS': 16,'KY': 17,'LA': 18,'ME': 19,'MD': 20,'MA': 21,'MI': 22,'MN': 23,'MS': 24,'MO': 25,'MT': 26,'NE': 27,'NV': 28,'NH': 29,'NJ': 30,'NM': 31,'NY': 32,'NC': 33,'ND': 34,'OH': 35,'OK': 36,'OR': 37,'PA': 38,'PR': 39,'RI': 40,'SC': 41,'SD': 42,'TN': 43,'TX': 44,'UT': 45,'VT': 46,'VA': 47,'WA': 48,'WV': 49,'WI': 50,'WY': 51})
  x_data['STATE'].astype('int64')

  #Adding Feature Discovery Month
  discovery_month = [];
  for i in range(len(x_data)):
   key = x_data.iloc[i]['DISCOVERY_DOY']
   if( 1 <= key <= 31 ):
    discovery_month.append(1)
   elif ( 32 <= key <= 60 ):
      discovery_month.append(2)
   elif ( 61 <= key <= 91 ):
     discovery_month.append(3)
   elif ( 92 <= key <= 121 ):
     discovery_month.append(4)
   elif ( 122 <= key <= 152 ):
     discovery_month.append(5)
   elif ( 153 <= key <= 182 ):
     discovery_month.append(6)
   elif ( 183 <= key <= 213 ):
     discovery_month.append(7)
   elif ( 214 <= key <= 244 ):
     discovery_month.append(8)
   elif ( 245 <= key <= 274 ):
     discovery_month.append(9)
   elif ( 275 <= key <= 305 ):
     discovery_month.append(10)
   elif ( 306 <= key <= 335 ):
     discovery_month.append(11)
   elif ( 336 <= key <= 366 ):
     discovery_month.append(12)

  x_data['DISCOVERY_MONTH'] = discovery_month
  x_data['DISCOVERY_MONTH'].astype('int64')

  #Delete DISCOVERY_DOY and CONT_DOY also now
  del x_data['DISCOVERY_DOY']
  del x_data['CONT_DOY']

  #Feature2 DISCOVERY_TOD
  discovery_tod = [];
  x_data['DISCOVERY_TIME'] = x_data['DISCOVERY_TIME'].replace([None],'0000')
  for i in range(len(x_data)):
    key = x_data.iloc[i]['DISCOVERY_TIME']
    if( key == '0000' ):
      discovery_tod.append(0)
    elif ( '0000' < key <= '0600' ):
      discovery_tod.append(1)
    elif ( '0600' < key <= '1200' ):
      discovery_tod.append(2)
    elif ( '1200' < key <= '1600' ):
      discovery_tod.append(3)
    elif ( '1600' < key <= '2000' ):
      discovery_tod.append(4)
    elif ( '2000' < key <= '2400' ):
      discovery_tod.append(5)

  x_data['DISCOVERY_TOD'] = discovery_tod
  x_data['DISCOVERY_TOD'].astype('int64')

  del x_data['DISCOVERY_TIME']

  x_data['LATITUDE'] = (x_data['LATITUDE']*10).apply(np.floor)/10
  x_data['LONGITUDE'] = (x_data['LONGITUDE']*10).apply(np.floor)/10

  #Add forest Area feature
  forest_Area = pd.read_excel('drive/My Drive/CaseStudy1/FOREST_Area.xlsx')
  forest_Area.head()
  STATE_PRCNT_FOREST = [];
  for i in range(len(x_data)):
    key = x_data.iloc[i]['STATE'].astype('int64')
    STATE_PRCNT_FOREST.append(forest_Area['Forest_Coverage'].values[key])

  x_data['STATE_PRCNT_FOREST'] = STATE_PRCNT_FOREST
  x_data['STATE_PRCNT_FOREST'].astype('float64')

  #Add Avg Temp Feature
  avg_temp  = pd.read_excel('drive/My Drive/CaseStudy1/avg_temp.xlsx')

  AVG_TEMP_LIST = [];
  for i in range(len(x_data)):
    state_key = x_data.iloc[i]['STATE'].astype('int64')
    year_key = x_data.iloc[i]['FIRE_YEAR'].astype('int64')
    AVG_TEMP_LIST.append(avg_temp[year_key].values[state_key])

  x_data['AVG_TEMP'] = AVG_TEMP_LIST
  x_data['AVG_TEMP'].astype('float64')

  #Add Avg Prec Feature
  avg_prec  = pd.read_excel('drive/My Drive/CaseStudy1/avg_prec.xlsx')

  AVG_PREC_LIST = [];
  for i in range(len(x_data)):
   state_key = x_data.iloc[i]['STATE'].astype('int64')
   AVG_PREC_LIST.append(avg_prec['Avg_Prec'].values[state_key])

  x_data['AVG_PREC'] = AVG_PREC_LIST
  x_data['AVG_PREC'].astype('float64')

  predictions = DataPrediction(x_data)

  #Got the prediction values, now computing the errors
  MAE_value = mean_absolute_error(y_data, predictions)
  print('Mean Absolute Error comes out to be: ', MAE_value, '\n')

  y_true, y_pred = np.array(y_data), np.array(predictions)
  MAPE_value = np.mean(np.abs((y_true - y_pred) / y_true)) * 100
  print('Mean Absolute Percentage Error is: ', MAPE_value)
```

In [19]:

```
function2(dataset)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  """
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   6 out of   6 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   7 out of   7 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   8 out of   8 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  10 out of  10 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  11 out of  11 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  12 out of  12 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  13 out of  13 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  14 out of  14 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  15 out of  15 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  16 out of  16 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  17 out of  17 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  18 out of  18 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  19 out of  19 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  20 out of  20 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  21 out of  21 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  22 out of  22 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  23 out of  23 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  24 out of  24 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  25 out of  25 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  26 out of  26 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  27 out of  27 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  28 out of  28 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  29 out of  29 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  31 out of  31 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  32 out of  32 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  33 out of  33 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  34 out of  34 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  35 out of  35 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  36 out of  36 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  37 out of  37 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  38 out of  38 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  39 out of  39 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  40 out of  40 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  41 out of  41 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  42 out of  42 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  43 out of  43 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  44 out of  44 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  45 out of  45 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  46 out of  46 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  47 out of  47 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  48 out of  48 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done  49 out of  49 | elapsed:    0.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 110 out of 110 | elapsed:    0.1s finished
Mean Absolute Error comes out to be:  0.5002222222222222

Mean Absolute Percentage Error is:  23.428306878306877
```

Mean Absolute Percentage Error (MAPE) is the most common error for Forcasting. In our study the MAPE value comes out to be 23.42% which means for the remaining ~ 77% of the times, the model is predicting the right firesize class.