Modeling Notebook One

This notebook contains the logic for performing Linear Regression, Ridge Regression, Random Forest Regression, and Random Forest Classification.

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

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

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

```
!ls
```

```
cleaned = pd.read_csv("/content/drive/Shareddrives/Data Science 303 Group Project/csv/cleaned_fires_data_with_four_closest_stations_nov21.csv")
cleaned = cleaned.replace([np.inf, -np.inf], np.nan)
cleaned = cleaned.dropna()
```

```
display(cleaned.head())
display(cleaned.info(verbose=True))

STATION_LIST = ["BODIE", "BROOKS", "COHASSET", "EEL_RIVER", "HELL_HOLE", "HERNANDEZ"]
STATION_LIST += ["HUNTER_MOUNTAIN", "JUANITA_LAKE", "LADDER_BUTTE", "LAS_TABLAS", "LA_HONDA", "OAK_CREEK"]
STATION_LIST += ["PANAMINT", "PILOT_HILL", "SCORPION", "SOLDIER_MOUNTAIN", "SQUAW_LAKE", "STAMPEDE", "VAN_BREMMER", "WOLVERTON"]
STATION_LIST = set(STATION_LIST)
```

	Unnamed: 0	BODIE	BROOKS	CORASSET	CONTAINMENT_MONTH	CONT_DATE	CONT_DOY	CONT_TIME	DISCOVERY_DATE	DISCOVERY_DOY	DISCOVERY_MONTH	DISCOVERY_TIME	EEL_RIVER	FIRE_SIZE	FIRE_SIZE_CLASS	FIRE_YEAR	FOD_ID	HELL_HOLE	HERNANDEZ	HUNTER_MOUNTAIN	JUANITA_LAKE	LADDER_BUTTE	LAS_TABLAS	LA_RONDA	NWCG_REPORTING_AGENCY	NWCG_REPORTING_UNIT_NAME	OAK_CREEK	OWNER_DESCR	PANAMINT	PILOT_HILL	PRIMARY_STATION_10_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH	PRIMARY_STATION_10_MONTHS_PRIOR_EXTREME_MINIMUM_TEMPERATURE_FOR_MON
0	0	-0.501752	-0.841998	-1.128433	2	2453403.5	33.0	1730.0	2453403.5	33	2	1300.0	-0.848324	0.10	A	2005	1	-1.000659	0.195610	0.434939	-0.886032	-1.114178	0.448456	-0.394781	FS	Plumas National Forest	0.297877	USFS	0.545114	-0.914818	-0.339638	-0.5000
1	1	-1.237103	-0.939629	-0.802799	5	2453137.5	133.0	1530.0	2453137.5	133	5	845.0	-0.597768	0.25	A	2004	2	-1.471873	-0.562875	-0.243079	-0.487813	-0.669480	-0.281374	-0.869095	FS	Eldorado National Forest	-0.451833	USFS	-0.057172	-1.280961	1.107121	1.2465
2	2	-1.070104	-1.059637	-0.891098	6	2453156.5	152.0	2024.0	2453156.5	152	6	1921.0	-0.695749	0.10	A	2004	3	-1.410920	-0.543260	-0.109734	-0.533929	-0.714976	-0.262071	-0.950563	FS	Eldorado National Forest	-0.317057	STATE OR PRIVATE	0.059122	-1.392229	0.449504	1.1794
3	3	-1.588084	-0.752356	-0.586044	7	2453189.5	185.0	1400.0	2453184.5	180	6	1600.0	-0.401895	0.10	A	2004	4	-1.213089	-0.755069	-0.564467	-0.322165	-0.486222	-0.477673	-0.836997	FS	Eldorado National Forest	-0.795000	USFS	-0.341413	-1.070171	0.054933	-1.1718
4	4	-1.578681	-0.759860	-0.590604	7	2453189.5	185.0	1200.0	2453184.5	180	6	1600.0	-0.407106	0.10	A	2004	5	-1.217813	-0.758604	-0.557924	-0.324379	-0.488626	-0.480494	-0.845502	FS	Eldorado National Forest	-0.788869	USFS	-0.335760	-1.077584	0.054933	-1.1718

5 rows × 255 columns

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 87106 entries, 0 to 87105
Data columns (total 255 columns):
 #   Column                                                                Dtype
---  ------                                                                -----
 0   Unnamed: 0                                                            int64
 1   BODIE                                                                 float64
 2   BROOKS                                                                float64
 3   CORASSET                                                              float64
 4   CONTAINMENT_MONTH                                                     int64
 5   CONT_DATE                                                             float64
 6   CONT_DOY                                                              float64
 7   CONT_TIME                                                             float64
 8   DISCOVERY_DATE                                                        float64
 9   DISCOVERY_DOY                                                         int64
 10  DISCOVERY_MONTH                                                       int64
 11  DISCOVERY_TIME                                                        float64
 12  EEL_RIVER                                                             float64
 13  FIRE_SIZE                                                             float64
 14  FIRE_SIZE_CLASS                                                       object
 15  FIRE_YEAR                                                             int64
 16  FOD_ID                                                                int64
 17  HELL_HOLE                                                             float64
 18  HERNANDEZ                                                             float64
 19  HUNTER_MOUNTAIN                                                       float64
 20  JUANITA_LAKE                                                          float64
 21  LADDER_BUTTE                                                          float64
 22  LAS_TABLAS                                                            float64
 23  LA_RONDA                                                              float64
 24  NWCG_REPORTING_AGENCY                                                 object
 25  NWCG_REPORTING_UNIT_NAME                                              object
 26  OAK_CREEK                                                             float64
 27  OWNER_DESCR                                                           object
 28  PANAMINT                                                              float64
 29  PILOT_HILL                                                            float64
 30  PRIMARY_STATION_10_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH     float64
 31  PRIMARY_STATION_10_MONTHS_PRIOR_EXTREME_MINIMUM_TEMPERATURE_FOR_MONTH float64
 32  PRIMARY_STATION_10_MONTHS_PRIOR_HEATING_DEGREE_DAYS_TO_DATE           float64
 33  PRIMARY_STATION_10_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY                float64
 34  PRIMARY_STATION_10_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY_CUMULATIVE     float64
 35  PRIMARY_STATION_10_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT float64
 36  PRIMARY_STATION_10_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT float64
 37  PRIMARY_STATION_10_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT float64
 38  PRIMARY_STATION_10_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_0_FAHRENHEIT  float64
 39  PRIMARY_STATION_10_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT float64
 40  PRIMARY_STATION_10_MONTHS_PRIOR_TEMPERATURE_AVERAGE                   float64
 41  PRIMARY_STATION_10_MONTHS_PRIOR_TEMPERATURE_MAX                       float64
 42  PRIMARY_STATION_10_MONTHS_PRIOR_TEMPERATURE_MIN                       float64
 43  PRIMARY_STATION_11_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH     float64
 44  PRIMARY_STATION_11_MONTHS_PRIOR_EXTREME_MINIMUM_TEMPERATURE_FOR_MONTH float64
 45  PRIMARY_STATION_11_MONTHS_PRIOR_HEATING_DEGREE_DAYS_TO_DATE           float64
 46  PRIMARY_STATION_11_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY                float64
 47  PRIMARY_STATION_11_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY_CUMULATIVE     float64
 48  PRIMARY_STATION_11_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT float64
 49  PRIMARY_STATION_11_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT float64
 50  PRIMARY_STATION_11_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT float64
 51  PRIMARY_STATION_11_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_0_FAHRENHEIT  float64
 52  PRIMARY_STATION_11_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT float64
 53  PRIMARY_STATION_11_MONTHS_PRIOR_TEMPERATURE_AVERAGE                   float64
 54  PRIMARY_STATION_11_MONTHS_PRIOR_TEMPERATURE_MAX                       float64
 55  PRIMARY_STATION_11_MONTHS_PRIOR_TEMPERATURE_MIN                       float64
 56  PRIMARY_STATION_12_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH     float64
 57  PRIMARY_STATION_12_MONTHS_PRIOR_EXTREME_MINIMUM_TEMPERATURE_FOR_MONTH float64
 58  PRIMARY_STATION_12_MONTHS_PRIOR_HEATING_DEGREE_DAYS_TO_DATE           float64
 59  PRIMARY_STATION_12_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY                float64
 60  PRIMARY_STATION_12_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY_CUMULATIVE     float64
 61  PRIMARY_STATION_12_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT float64
 62  PRIMARY_STATION_12_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT float64
 63  PRIMARY_STATION_12_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT float64
 64  PRIMARY_STATION_12_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_0_FAHRENHEIT  float64
 65  PRIMARY_STATION_12_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT float64
 66  PRIMARY_STATION_12_MONTHS_PRIOR_TEMPERATURE_AVERAGE                   float64
 67  PRIMARY_STATION_12_MONTHS_PRIOR_TEMPERATURE_MAX                       float64
 68  PRIMARY_STATION_12_MONTHS_PRIOR_TEMPERATURE_MIN                       float64
 69  PRIMARY_STATION_13_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH     float64
 70  PRIMARY_STATION_13_MONTHS_PRIOR_EXTREME_MINIMUM_TEMPERATURE_FOR_MONTH float64
 71  PRIMARY_STATION_13_MONTHS_PRIOR_HEATING_DEGREE_DAYS_TO_DATE           float64
 72  PRIMARY_STATION_13_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY                float64
 73  PRIMARY_STATION_13_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY_CUMULATIVE     float64
 74  PRIMARY_STATION_13_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT float64
 75  PRIMARY_STATION_13_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT float64
 76  PRIMARY_STATION_13_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT float64
 77  PRIMARY_STATION_13_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_0_FAHRENHEIT  float64
 78  PRIMARY_STATION_13_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT float64
 79  PRIMARY_STATION_13_MONTHS_PRIOR_TEMPERATURE_AVERAGE                   float64
 80  PRIMARY_STATION_13_MONTHS_PRIOR_TEMPERATURE_MAX                       float64
 81  PRIMARY_STATION_13_MONTHS_PRIOR_TEMPERATURE_MIN                       float64
 82  PRIMARY_STATION_14_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH     float64
 83  PRIMARY_STATION_14_MONTHS_PRIOR_EXTREME_MINIMUM_TEMPERATURE_FOR_MONTH float64
 84  PRIMARY_STATION_14_MONTHS_PRIOR_HEATING_DEGREE_DAYS_TO_DATE           float64
 85  PRIMARY_STATION_14_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY                float64
 86  PRIMARY_STATION_14_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY_CUMULATIVE     float64
 87  PRIMARY_STATION_14_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT float64
 88  PRIMARY_STATION_14_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT float64
 89  PRIMARY_STATION_14_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT float64
 90  PRIMARY_STATION_14_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_0_FAHRENHEIT  float64
 91  PRIMARY_STATION_14_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT float64
 92  PRIMARY_STATION_14_MONTHS_PRIOR_TEMPERATURE_AVERAGE                   float64
 93  PRIMARY_STATION_14_MONTHS_PRIOR_TEMPERATURE_MAX                       float64
 94  PRIMARY_STATION_14_MONTHS_PRIOR_TEMPERATURE_MIN                       float64
 95  PRIMARY_STATION_15_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH     float64
 96  PRIMARY_STATION_15_MONTHS_PRIOR_EXTREME_MINIMUM_TEMPERATURE_FOR_MONTH float64
 97  PRIMARY_STATION_15_MONTHS_PRIOR_HEATING_DEGREE_DAYS_TO_DATE           float64
 98  PRIMARY_STATION_15_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY                float64
 99  PRIMARY_STATION_15_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY_CUMULATIVE     float64
 100 PRIMARY_STATION_15_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT float64
 101 PRIMARY_STATION_15_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT float64
 102 PRIMARY_STATION_15_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT float64
 103 PRIMARY_STATION_15_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_0_FAHRENHEIT  float64
 104 PRIMARY_STATION_15_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT float64
 105 PRIMARY_STATION_15_MONTHS_PRIOR_TEMPERATURE_AVERAGE                   float64
 106 PRIMARY_STATION_15_MONTHS_PRIOR_TEMPERATURE_MAX                       float64
 107 PRIMARY_STATION_15_MONTHS_PRIOR_TEMPERATURE_MIN                       float64
 108 PRIMARY_STATION_16_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH     float64
 109 PRIMARY_STATION_16_MONTHS_PRIOR_EXTREME_MINIMUM_TEMPERATURE_FOR_MONTH float64
 110 PRIMARY_STATION_16_MONTHS_PRIOR_HEATING_DEGREE_DAYS_TO_DATE           float64
 111 PRIMARY_STATION_16_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY                float64
 112 PRIMARY_STATION_16_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY_CUMULATIVE     float64
 113 PRIMARY_STATION_16_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT float64
 114 PRIMARY_STATION_16_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT float64
 115 PRIMARY_STATION_16_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT float64
 116 PRIMARY_STATION_16_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_0_FAHRENHEIT  float64
 117 PRIMARY_STATION_16_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT float64
 118 PRIMARY_STATION_16_MONTHS_PRIOR_TEMPERATURE_AVERAGE                   float64
 119 PRIMARY_STATION_16_MONTHS_PRIOR_TEMPERATURE_MAX                       float64
 120 PRIMARY_STATION_16_MONTHS_PRIOR_TEMPERATURE_MIN                       float64
 121 PRIMARY_STATION_1_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH      float64
 122 PRIMARY_STATION_1_MONTHS_PRIOR_EXTREME_MINIMUM_TEMPERATURE_FOR_MONTH  float64
 123 PRIMARY_STATION_1_MONTHS_PRIOR_HEATING_DEGREE_DAYS_TO_DATE            float64
 124 PRIMARY_STATION_1_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY                 float64
 125 PRIMARY_STATION_1_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY_CUMULATIVE      float64
 126 PRIMARY_STATION_1_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT float64
 127 PRIMARY_STATION_1_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT float64
 128 PRIMARY_STATION_1_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT float64
 129 PRIMARY_STATION_1_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_0_FAHRENHEIT  float64
```

```
129 PRIMARY_STATION_1_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_0_FAHRENHEIT    float64
130 PRIMARY_STATION_1_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT   float64
131 PRIMARY_STATION_1_MONTHS_PRIOR_TEMPERATURE_AVERAGE                          float64
132 PRIMARY_STATION_1_MONTHS_PRIOR_TEMPERATURE_MAX                              float64
133 PRIMARY_STATION_1_MONTHS_PRIOR_TEMPERATURE_MIN                              float64
134 PRIMARY_STATION_2_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH            float64
135 PRIMARY_STATION_2_MONTHS_PRIOR_EXTREME_MINIMUM_TEMPERATURE_FOR_MONTH       float64
136 PRIMARY_STATION_2_MONTHS_PRIOR_HEATING_DEGREE_DAYS_TO_DATE                  float64
137 PRIMARY_STATION_2_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY                       float64
138 PRIMARY_STATION_2_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY_CUMULATIVE           float64
139 PRIMARY_STATION_2_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT float64
140 PRIMARY_STATION_2_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT  float64
141 PRIMARY_STATION_2_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT  float64
142 PRIMARY_STATION_2_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_0_FAHRENHEIT   float64
143 PRIMARY_STATION_2_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT  float64
144 PRIMARY_STATION_2_MONTHS_PRIOR_TEMPERATURE_AVERAGE                          float64
145 PRIMARY_STATION_2_MONTHS_PRIOR_TEMPERATURE_MAX                              float64
146 PRIMARY_STATION_2_MONTHS_PRIOR_TEMPERATURE_MIN                              float64
147 PRIMARY_STATION_3_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH            float64
```

Final Part of Normalization: MinMax scaling latitude and longitude

```
1 from sklearn.preprocessing import MinMaxScaler
2 scaler = MinMaxScaler()
3
4 lat = scaler.fit_transform(np.array(cleaned["LATITUDE"]).reshape(-1, 1))
5 long = scaler.fit_transform(np.array(cleaned["LONGITUDE"]).reshape(-1, 1))
6
7 # DROP ALL LATITUDE LONGITUDE COLUMNS, THEN READD THE SCALED LATITUDE AND LONGITUDE
8 drops = []
9 for col in cleaned.columns:
10   if "LATITUDE" in col or "LONGITUDE" in col:
11     drops.append(col)
12
13 for col in drops:
14   del cleaned[col]
15
16 cleaned["S_LATITUDE"] = lat
17 cleaned["S_LONGITUDE"] = long
18
```

```
172 PRIMARY_STATION_4_MONTHS_PRIOR_TEMPERATURE_MIN                             float64
```

```
1 # Elevation for the primary station is redundant. Drop all elevation columns from the primary station, except for one
2 elevation = cleaned["PRIMARY_STATION_1_MONTHS_PRIOR_ELEVATION"]
3
4 drops = []
5 for col in cleaned.columns:
6   if "PRIMARY_STATION" in col and "ELEVATION" in col:
7     drops.append(col)
8
9 for col in drops:
10   del cleaned[col]
11
12 cleaned["PRIMARY_STATION_ELEVATION"] = elevation
13
```

```
188 PRIMARY_STATION_6_MONTHS_PRIOR_HEATING_DEGREE_DAYS_TO_DATE                 float64
```

```
1 # We should have a large number of fires. If not, fail
2 print(len(cleaned))
3 assert(len(cleaned) > 70000)
```

```
87106
```

```
1 print( (1 - 360) % 365)
```

```
6
```

```
499 PRIMARY_STATION_7_MONTHS_PRIOR_HEATING_DEGREE_DAYS_TO_DATE                 float64
```

```
1 cleaned['DURATION'] = (cleaned['CONT_DOY']-cleaned['DISCOVERY_DOY'])%365
```

```
452 PRIMARY_STATION_7_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY_CUMULATIVE          float64
```

```
1 cleaned['DURATION'].max()
```

```
335.0
```

```
1 from sklearn.linear_model import LinearRegression, Ridge
```

```
411 PRIMARY_STATION_7_MONTHS_PRIOR_TEMPERATURE_MIN                            float64
```

```
1 # Working off of closest_station_1 for now, we can replicate later with other stations if we want
2 # 1. Get number of days that the fire was going (containment date - discovery date)
3 # Model 1a: fire duration
```

```
414 PRIMARY_STATION_8_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY_CUMULATIVE          float64
```

```
1 numerical = cleaned.select_dtypes(include="number")
2 numerical.head()
3 numerical.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 87106 entries, 0 to 87105
Data columns (total 246 columns):
 #   Column                                                        Dtype
---  ------                                                        -----
 0   Unnamed: 0                                                    int64
 1   BODIE                                                         float64
 2   BROOKS                                                        float64
 3   COBASSET                                                      float64
 4   CONTAINMENT_MONTH                                             int64
 5   CONT_DATE                                                     float64
 6   CONT_DOY                                                      float64
 7   CONT_TIME                                                     float64
 8   DISCOVERY_DATE                                                float64
 9   DISCOVERY_DOY                                                 int64
 10  DISCOVERY_MONTH                                               int64
 11  DISCOVERY_TIME                                                float64
 12  EEL_RIVER                                                     float64
 13  FIRE_SIZE                                                     float64
 14  FIRE_YEAR                                                     int64
 15  FOD_ID                                                        int64
 16  HELL_HOLE                                                     float64
 17  HERNANDEZ                                                     float64
 18  HUNTER_MOUNTAIN                                               float64
 19  JUANITA_LAKE                                                  float64
 20  LADDER_BUTTE                                                  float64
 21  LAS_TABLAS                                                    float64
 22  LA_HONDA                                                      float64
 23  OAK_CREEK                                                     float64
 24  PANAMINT                                                      float64
 25  PILOT_HILL                                                    float64
 26  PRIMARY_STATION_10_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH  float64
 27  PRIMARY_STATION_10_MONTHS_PRIOR_EXTREME_MINIMUM_TEMPERATURE_FOR_MONTH  float64
 28  PRIMARY_STATION_10_MONTHS_PRIOR_HEATING_DEGREE_DAYS_TO_DATE   float64
 29  PRIMARY_STATION_10_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY        float64
 30  PRIMARY_STATION_10_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY_CUMULATIVE  float64
 31  PRIMARY_STATION_10_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT  float64
 32  PRIMARY_STATION_10_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT  float64
 33  PRIMARY_STATION_10_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT  float64
 34  PRIMARY_STATION_10_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_0_FAHRENHEIT   float64
 35  PRIMARY_STATION_10_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT  float64
 36  PRIMARY_STATION_10_MONTHS_PRIOR_TEMPERATURE_AVERAGE           float64
 37  PRIMARY_STATION_10_MONTHS_PRIOR_TEMPERATURE_MAX               float64
 38  PRIMARY_STATION_10_MONTHS_PRIOR_TEMPERATURE_MIN               float64
 39  PRIMARY_STATION_11_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH  float64
 40  PRIMARY_STATION_11_MONTHS_PRIOR_EXTREME_MINIMUM_TEMPERATURE_FOR_MONTH  float64
 41  PRIMARY_STATION_11_MONTHS_PRIOR_HEATING_DEGREE_DAYS_TO_DATE   float64
 42  PRIMARY_STATION_11_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY        float64
 43  PRIMARY_STATION_11_MONTHS_PRIOR_NUM_COOLING_DEGREE_DAY_CUMULATIVE  float64
 44  PRIMARY_STATION_11_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT  float64
 45  PRIMARY_STATION_11_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT  float64
 46  PRIMARY_STATION_11_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT  float64
 47  PRIMARY_STATION_11_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_0_FAHRENHEIT   float64
 48  PRIMARY_STATION_11_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT  float64
 49  PRIMARY_STATION_11_MONTHS_PRIOR_TEMPERATURE_AVERAGE           float64
 50  PRIMARY_STATION_11_MONTHS_PRIOR_TEMPERATURE_MAX               float64
```

```
51  PRIMARY_STATION_11_MONTHS_PRIOR_TEMPERATURE_MIN                    float64
```

```
cols = numerical.columns
closest = []

for col in cols:
  if (col[:16] == 'CLOSEST_STATION_' or col in STATION_LIST or col == "S_LATITUDE" or col == "S_LONGITUDE"):
    closest.append(col)
```

```
primary_station_data_cols = []
for col in cols:
  if ("PRIMARY_STATION" in col or col in STATION_LIST or col == "S_LATITUDE" or col == "S_LONGITUDE"):
    primary_station_data_cols.append(col)

primary_station_df = pd.DataFrame()
for col in primary_station_data_cols:
  primary_station_df[col] = numerical[col]

primary_station_df["FIRE_SIZE"] = numerical["FIRE_SIZE"]
primary_station_df["DURATION"] = numerical["DURATION"]

```

```
NUM_FEATURES_TO_KEEP = 300 + 2
correlation = primary_station_df.corr(method='pearson')
highest_correlation = (correlation.nlargest(NUM_FEATURES_TO_KEEP, 'FIRE_SIZE').index)
print(f"TOP {NUM_FEATURES_TO_KEEP} FEATURES WITH HIGHEST CORRELATION TO FIRE SIZE")
print(list(highest_correlation))

print(f"TOP 10 FEATURES WITH HIGHEST CORRELATION TO FIRE SIZE")
display(highest_correlation[2:12])
display(correlation.nlargest(NUM_FEATURES_TO_KEEP, 'FIRE_SIZE')[2:12]["FIRE_SIZE"])

fire_size_prediction_df = primary_station_df[highest_correlation]
del fire_size_prediction_df["FIRE_SIZE"]
del fire_size_prediction_df["DURATION"]
```

```
TOP 302 FEATURES WITH HIGHEST CORRELATION TO FIRE SIZE
['FIRE_SIZE', 'DURATION', 'PRIMARY_STATION_13_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH', 'PRIMARY_STATION_1_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH', 'PRIMARY_STATION_13_MONTHS_PRIOR_TEMPERATURE_MAX', 'PRIMARY_STATION_1_MONTHS_PRIOR_TEMPERATURE_MAX', 'PRIMARY_STATION_12_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT', 'PRIMARY_STATION_12_MONTHS_PRIOR_NUM_DAYS_WI
TOP 10 FEATURES WITH HIGHEST CORRELATION TO FIRE SIZE
Index(['PRIMARY_STATION_13_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH',
       'PRIMARY_STATION_1_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH',
       'PRIMARY_STATION_13_MONTHS_PRIOR_TEMPERATURE_MAX',
       'PRIMARY_STATION_1_MONTHS_PRIOR_TEMPERATURE_MAX',
       'PRIMARY_STATION_12_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT',
       'PRIMARY_STATION_12_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT',
       'PRIMARY_STATION_13_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT',
       'PRIMARY_STATION_1_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT',
       'PRIMARY_STATION_12_MONTHS_PRIOR_TEMPERATURE_MAX',
       'PRIMARY_STATION_12_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH'],
      dtype='object')
PRIMARY_STATION_13_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH            0.022363
PRIMARY_STATION_1_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH             0.022363
PRIMARY_STATION_13_MONTHS_PRIOR_TEMPERATURE_MAX                              0.020201
PRIMARY_STATION_1_MONTHS_PRIOR_TEMPERATURE_MAX                               0.020201
PRIMARY_STATION_12_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT   0.019643
PRIMARY_STATION_12_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT   0.019516
PRIMARY_STATION_13_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT   0.019503
PRIMARY_STATION_1_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT    0.019503
PRIMARY_STATION_12_MONTHS_PRIOR_TEMPERATURE_MAX                              0.019122
PRIMARY_STATION_12_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH            0.018295
Name: FIRE_SIZE, dtype: float64
```

```
display(fire_size_prediction_df)
```

	PRIMARY_STATION_13_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH	PRIMARY_STATION_1_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH	PRIMARY_STATION_13_MONTHS_PRIOR_TEMPERATURE_MAX	PRIMARY_STATION_1_MONTHS_PRIOR_TEMPERATURE_MAX	PRIMARY_STATION_12_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_70_FAHRENHEIT	PRIMARY_STATION_12_MONTHS_PRIOR_NUM_DAYS_WITH_MAX_TEMP_ABOVE_90_FAHRENHEIT
0	-0.931493	-0.931493	-1.345868	-1.345868	-1.159138	-0.641117
1	-0.997255	-0.997255	-1.277944	-1.277944	0.053157	-0.143038
2	0.515265	0.515265	-0.092363	-0.092363	0.942172	-0.641117
3	-0.471161	-0.471161	-0.283785	-0.283785	0.780533	-0.641117
4	-0.471161	-0.471161	-0.283785	-0.283785	0.780533	-0.641117
...	...	...	...	...	...	...
87101	0.712551	0.712551	0.722723	0.722723	1.265451	2.265798
87102	1.370168	1.370168	1.395787	1.395787	0.942172	1.412221
87103	1.698977	1.698977	1.735407	1.735407	1.346270	2.265798
87104	0.317980	0.317980	-0.129413	-0.129413	-0.916679	-0.641117
87105	1.370168	1.370168	1.395787	1.395787	0.942172	1.412221

87106 rows × 231 columns

```
NUM_FEATURES_TO_KEEP = 300 + 2
correlation = primary_station_df.corr(method='pearson')
highest_correlation = (correlation.nlargest(NUM_FEATURES_TO_KEEP, 'DURATION').index)
print(f"TOP {NUM_FEATURES_TO_KEEP} FEATURES WITH HIGHEST CORRELATION TO FIRE DURATION")
display(highest_correlation)

print(f"TOP 10 FEATURES WITH HIGHEST CORRELATION TO FIRE DURATION")
display(highest_correlation[2:12])
display(correlation.nlargest(NUM_FEATURES_TO_KEEP, 'DURATION')[2:12]["DURATION"])

fire_duration_prediction_df = primary_station_df[highest_correlation]
del fire_duration_prediction_df["FIRE_SIZE"]
del fire_duration_prediction_df["DURATION"]
```

```
TOP 302 FEATURES WITH HIGHEST CORRELATION TO FIRE DURATION
Index(['DURATION', 'FIRE_SIZE',
       'PRIMARY_STATION_6_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT',
       'PRIMARY_STATION_6_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT',
       'PRIMARY_STATION_7_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT',
       'PRIMARY_STATION_ELEVATION',
       'PRIMARY_STATION_5_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT',
       'PRIMARY_STATION_5_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT',
       'PRIMARY_STATION_7_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT',
       'PRIMARY_STATION_12_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT',
```

```
display(fire_duration_prediction_df)
```

	PRIMARY_STATION_6_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT	PRIMARY_STATION_6_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT	PRIMARY_STATION_7_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_TEMP_BELOW_65_FAHRENHEIT	PRIMARY_STATION_ELEVATION	PRIMARY_STATION_5_MONTHS_PRIOR_NUM_DAYS_WITH_MIN_TEMP_BELOW_32_FAHRENHEIT	PRIMARY_STATION_5_MONTHS_PRIOR_NUM_DAYS_WHERE_AVG_T
0	-0.869653	-0.935826	-0.995396	1.192259	0.394730	
1	1.227107	1.291149	-0.737623	0.525056	1.291149	
2	1.506884	1.291149	1.227107	0.525056	1.243994	
3	2.613420	1.676955	2.229905	1.808604	1.740558	
4	2.613420	1.676955	2.229905	1.808604	1.740558	
...	...	...	...	...	...	
87101	0.548096	0.394730	0.133145	-1.008744	0.837331	
87102	-0.036608	-0.028466	0.218021	0.180489	0.394730	
87103	-0.055469	-0.232580	0.425497	-1.008744	-0.232580	
87104	-0.391831	-0.935826	-0.844505	-1.266260	-0.505710	
87105	-0.036608	-0.028466	0.218021	0.180489	0.394730	

87106 rows × 231 columns

```
PRIMARY_STATION_ELEVATION                                    0.083869
```

```
closest_df = pd.DataFrame()

for col in closest:
  closest_df[col] = numerical[col]
```

```
closest_df.head()
```

	BODIE	BROOKS	CONNASSET	EEL_RIVER	HELL_HOLE	HERNANDES	HUNTER_MOUNTAIN	JUANITA_LAKE	LADDER_BUTTE	LAS_TABLAS	LA_HONDA	OAK_CREEK	PANAMINT	PILOT_HILL	SCORPION	SOLDIER_MOUNTAIN	SQUAW_LAKE	STAMPEDE	VAN_BREMMER	WOLVERTON	S_LATITUDE	S_LONGITUDE
0	-0.501752	-0.841998	-1.128433	-0.848324	-1.000659	0.195610	0.434939	-0.886032	-1.114178	0.448456	-0.394781	0.297877	0.545114	-0.914818	-0.846454	-1.044163	0.687200	-1.139269	-0.918627	0.410736	0.788242	0.330313
1	-1.237103	-0.939629	-0.802799	-0.597768	-1.471873	-0.562875	-0.243079	-0.487813	-0.669480	-0.281374	-0.869095	-0.451833	-0.057172	-1.280961	-0.473194	-0.612901	0.315528	-1.287797	-0.513504	-0.311271	0.672217	0.388850
2	-1.070104	-1.059637	-0.891098	-0.695749	-1.410920	-0.543260	-0.109734	-0.533929	-0.714976	-0.262071	-0.950563	-0.317057	0.059122	-1.392229	-0.537641	-0.663661	0.382606	-1.233892	-0.557916	-0.204482	0.677589	0.356621
3	-1.588084	-0.752356	-0.586044	-0.401895	-1.213089	-0.755069	-0.564467	-0.322165	-0.486222	-0.477673	-0.836997	-0.795000	-0.341413	-1.070171	-0.296636	-0.431126	0.143227	-1.125369	-0.346639	-0.618217	0.632919	0.436654
4	-1.578681	-0.759860	-0.590604	-0.407106	-1.217813	-0.758604	-0.557924	-0.324379	-0.488626	-0.480494	-0.845502	-0.788869	-0.335760	-1.077584	-0.299859	-0.433722	0.146406	-1.126335	-0.348787	-0.614275	0.632919	0.434734

Model 1: Linear Regression

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, median_absolute_error, r2_score
from sklearn.metrics import explained_variance_score
from sklearn import svm
from sklearn.preprocessing import PowerTransformer
from sklearn.ensemble import RandomForestRegressor

# def log_transform(column):
#   """
#   Transforms a feature so that it is scaled logarithmically. Useful for correcting floating point errors
#   """


# def min_max_transform(column):
#   """
#   Transforms a feature using min-max scaling
#   """
#   pass

def run_full_linear_regression_with_accuracy(features, response, n_highest = 0):
  print(f"Response Minimum: {response.min()}")
  print(f"Response Maximum: {response.max()}")

  model = LinearRegression()
  # Split into test and training set
  X_train, X_test, y_train, y_test = train_test_split(features, response, test_size=0.2, random_state=2020)
  model.fit(X_train, y_train)

  training_predictions = model.predict(X_train)
  test_predictions = model.predict(X_test)
  print(f"Linear Regression Coefficients: { model.coef_ }\nLinear Regression Intercept: { model.intercept_}\n")

  print("Training: mean absolute error: ", mean_absolute_error(y_train, training_predictions))
  print("Test mean absolute error: ", mean_absolute_error(y_test, test_predictions))

  print("Training average error rate: ", median_absolute_error(y_train, training_predictions))
  print("Test average error rate: ", median_absolute_error(y_test, test_predictions))

  print("Explained variance in training set is: ", explained_variance_score(y_train, training_predictions))
  print("Explained variance in test set is: ", explained_variance_score(y_test, test_predictions))

  print("R2 Score training set: ", r2_score(y_train, training_predictions))
  print("R2 Score testing set: ", r2_score(y_test, test_predictions))

  fig = plt.figure()
  ax1 = fig.add_subplot()
  ax1.set_xlabel("True Value from Test Set")
  ax1.set_ylabel('Prediction from Test Set')
  ax1.set_title('True Value vs Predicted Value for Test Set: Linear Regression')
  ax1.scatter(y_test, test_predictions)
  plt.show()

  # Residual plot against predictor
  fig = plt.figure()
  ax1 = fig.add_subplot()
  ax1.set_xlabel("Test Set Prediction")
  ax1.set_ylabel('Residuals from Test Set')
  ax1.set_title('Residual Graph: Linear Regression')
  ax1.scatter(test_predictions, y_test - test_predictions)
  plt.show()


def run_full_ridge_regression_with_accuracy(features, response, n_highest = 0):
  print(f"Response Minimum: {response.min()}")
  print(f"Response Maximum: {response.max()}")


  model = Ridge()
  # Split into test and training set
  X_train, X_test, y_train, y_test = train_test_split(features, response, test_size=0.2, random_state=2020)
  model.fit(X_train, y_train)

  training_predictions = model.predict(X_train)
  test_predictions = model.predict(X_test)
  print(f"Ridge Regression Coefficients: { model.coef_ }\Ridge Regression Intercept: { model.intercept_}\n")
```

```

    print("Training: mean absolute error: ", mean_absolute_error(y_train, training_predictions))
    print("Test mean absolute error: ", mean_absolute_error(y_test, test_predictions))

    print("Training average error rate: ", median_absolute_error(y_train, training_predictions))
    print("Test average error rate: ", median_absolute_error(y_test, test_predictions))

    print("Explained variance in training set is: ", explained_variance_score(y_train, training_predictions))
    print("Explained variance in test set is: ", explained_variance_score(y_test, test_predictions))

    print("R2 Score training set: ", r2_score(y_train, training_predictions))
    print("R2 Score testing set: ", r2_score(y_test, test_predictions))

    fig = plt.figure()
    ax1 = fig.add_subplot()
    ax1.set_xlabel("True Value from Test Set")
    ax1.set_ylabel('Prediction from Test Set')
    ax1.set_title('True Value vs Predicted Value for Test Set: Ridge Regression')
    ax1.scatter(y_test, test_predictions)
    plt.show()

    # Residual plot against predictor
    fig = plt.figure()
    ax1 = fig.add_subplot()
    ax1.set_xlabel("Test Set Prediction")
    ax1.set_ylabel('Residuals from Test Set')
    ax1.set_title('Residual Graph: Ridge Regression')
    ax1.scatter(test_predictions, y_test - test_predictions)
    plt.show()

def run_full_SVM_regression_with_accuracy(features, response, n_highest = 0):
    print(f"Response Minimum: {response.min()}")
    print(f"Response Maximum: {response.max()}")


    model = svm.SVR()
    # Split into test and training set
    X_train, X_test, y_train, y_test = train_test_split(features, response, test_size=0.2, random_state=2020)
    model.fit(X_train, y_train)

    training_predictions = model.predict(X_train)
    test_predictions = model.predict(X_test)

    print("Training: mean absolute error: ", mean_absolute_error(y_train, training_predictions))
    print("Test mean absolute error: ", mean_absolute_error(y_test, test_predictions))

    print("Training average error rate: ", median_absolute_error(y_train, training_predictions))
    print("Test average error rate: ", median_absolute_error(y_test, test_predictions))

    print("Explained variance in training set is: ", explained_variance_score(y_train, training_predictions))
    print("Explained variance in test set is: ", explained_variance_score(y_test, test_predictions))

    fig = plt.figure()
    ax1 = fig.add_subplot()
    ax1.set_xlabel("True Value from Test Set")
    ax1.set_ylabel('Prediction from Test Set')
    ax1.set_title('True Value vs Predicted Value for Test Set: SVM Regression')
    ax1.scatter(y_test, test_predictions)
    plt.show()

    # Residual plot against predictor
    fig = plt.figure()
    ax1 = fig.add_subplot()
    ax1.set_xlabel("Test Set Prediction")
    ax1.set_ylabel('Residuals from Test Set')
    ax1.set_title('Residual Graph: SVM Regression')
    ax1.scatter(test_predictions, y_test - test_predictions)
    plt.show()

def run_full_random_forest_regression_with_accuracy(features, response, n_highest = 0):
    print(f"Response Minimum: {response.min()}")
    print(f"Response Maximum: {response.max()}")


    model = RandomForestRegressor(n_estimators=100)
    # Split into test and training set
    X_train, X_test, y_train, y_test = train_test_split(features, response, test_size=0.2, random_state=2020)
    model.fit(X_train, y_train)

    training_predictions = model.predict(X_train)
    test_predictions = model.predict(X_test)

    print("Training: mean absolute error: ", mean_absolute_error(y_train, training_predictions))
    print("Test mean absolute error: ", mean_absolute_error(y_test, test_predictions))

    print("Training average error rate: ", median_absolute_error(y_train, training_predictions))
    print("Test average error rate: ", median_absolute_error(y_test, test_predictions))

    print("Explained variance in training set is: ", explained_variance_score(y_train, training_predictions))
    print("Explained variance in test set is: ", explained_variance_score(y_test, test_predictions))

    fig = plt.figure()
    ax1 = fig.add_subplot()
    ax1.set_xlabel("True Value from Test Set")
    ax1.set_ylabel('Prediction from Test Set')
    ax1.set_title('True Value vs Predicted Value for Test Set: Random Forest Regression')
    ax1.scatter(y_test, test_predictions)
    plt.show()

    fig = plt.figure()

    ax1 = fig.add_subplot()
    ax1.set_xlabel("True Value from Training Set")
    ax1.set_ylabel('Prediction from Training Set')
    ax1.set_title('True Value vs Predicted Value for Training Set: Random Forest Regression')
    ax1.scatter(y_train, training_predictions)
    plt.show()

    # Residual plot against predictor
    fig = plt.figure()
    ax1 = fig.add_subplot()
    ax1.set_xlabel("Test Set Prediction")
    ax1.set_ylabel('Residuals from Test Set')
    ax1.set_title('Residual Graph: Random Forest')
    ax1.scatter(test_predictions, y_test - test_predictions)
    plt.show()
```

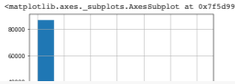
```
SUBSET = -1
```

Fire Size Prediction using Linear Regression and Ridge Regression

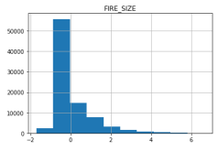
Here, we predict the size of a fire using linear regression

```
FIRE_SIZE = numerical["FIRE_SIZE"]
FIRE_SIZE.hist()
```

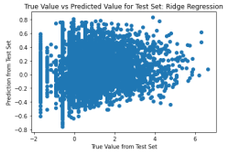
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5d993f86d0>
```

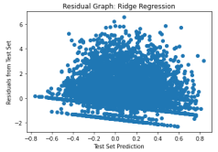


```
log_fire_size = pd.DataFrame(np.log(FIRE_SIZE))
log_fire_size = log_fire_size.replace([np.inf, -np.inf], np.nan)
log_fire_size = log_fire_size.fillna(0)
log_fire_size = (log_fire_size - log_fire_size.mean()) / log_fire_size.std()
log_fire_size.hist()
plt.show()

# run_full_linear_regression_with_accuracy(fire_size_prediction_df[:SUBSET], log_fire_size[:SUBSET])
run_full_ridge_regression_with_accuracy(fire_size_prediction_df[:SUBSET], log_fire_size[:SUBSET])
```




```
Response Minimum: FIRE_SIZE   -1.698422
dtype: float64
Response Maximum: FIRE_SIZE    6.638909
dtype: float64
Ridge Regression Coefficients: [[ 4.33594698e-02  4.33594698e-02  1.03918029e-01  1.03918029e-01
  -3.55139592e-02  1.94372343e-02 -2.37730649e-03 -2.37730649e-03
   2.95197670e-01 -5.50421501e-03 -1.64275460e-02 -1.64275460e-02
   2.09743454e-02  2.09743454e-02 -2.73514574e-03 -2.73514574e-03
  -2.32735103e-01 -2.32735103e-01  5.67128831e-02  1.03293300e-01
   1.03293300e-01 -3.10742784e-02 -9.37498845e-02 -1.59790510e-01
  -8.65886988e-03 -2.02195149e-03 -2.02195149e-03  1.63489531e-01
   1.63489531e-01  4.85216754e-02 -7.13064955e-03 -7.13064954e-03
  -1.53354316e-01 -1.53354316e-01 -1.89397625e-03 -1.89397625e-03
   1.25155578e-01  1.25155780e-01 -3.62207874e-02 -3.62207874e-02
   3.87649977e-02  3.87649977e-02 -2.95336422e-01 -2.95336422e-01
  -7.49515622e-03 -7.49515622e-03 -1.50014523e-02 -1.50014523e-02
   4.71990775e-02  1.57950987e-02  1.57950987e-02  3.95388936e-02
   3.95388936e-02 -2.96773002e-01  2.17586250e+00 -6.20496107e-01
   8.02527227e-03  8.02527227e-03  1.92007303e-02  1.92007303e-02
  -1.21936264e+00 -1.25427278e-01  3.96608944e-02  3.96608944e-02
   2.29678042e-02  2.29678042e-02  8.13495083e-02  8.13495083e-02
  -2.46745638e-02 -2.46745638e-02  1.22420091e-02 -2.83519348e-02
  -1.29031438e+00 -4.49127615e-02  1.43418219e+00  4.66073026e-02
  -1.64123986e+00  6.40358408e-03  6.40358408e-03  1.08638129e+00
  -4.82500336e-01 -1.10822859e-02  3.25933961e-01  1.96258308e-01
   3.96117084e-02 -5.51759322e-02 -5.51759323e-02 -4.70122684e-03
  -4.70122683e-03  1.17971539e-02  1.17971539e-02 -1.06617077e-02
  -1.06617077e-02 -2.67885150e-02  1.70111535e-03  5.34686540e-02
  -1.99321352e-02 -1.99321352e-02  9.68996125e-02  1.97705340e-02
   1.08665133e+00 -3.54806340e-02 -3.58292019e-02 -2.88611820e-02
  -6.10815315e-02  2.84367189e-02  2.84367189e-02 -1.93343131e-02
  -1.93343131e-02  2.27118163e-02 -6.853555161e-04 -6.853555161e-04
   1.43410472e-02 -7.64684468e-02 -3.35278327e-02  4.95580216e-02
   3.39656156e+00  1.31859290e-01 -2.77583729e-02 -1.97469144e-02
  -7.52814716e-03  1.16471480e-02 -7.92131730e-02 -3.61492458e-02
   6.83011830e-03 -6.34140872e-02 -2.91079150e-03 -3.86656895e-02
   8.66979561e-03  1.46205124e-01 -5.60352074e-02  9.68871324e-01
  -9.80762352e-02  3.17596654e-02 -4.23048647e-02  1.02074299e-03
  -4.34179526e-02 -7.49110514e-03 -7.49110514e-03  2.83423971e-02
   5.44439329e-02  7.70594034e-02 -5.35758603e-01  2.60590537e-01
  -3.45819843e-02 -6.60734163e-03 -1.77984248e-02 -1.77984248e-02
   6.428048407e-02  2.33412963e-03 -4.30558901e-01 -7.29125279e-01
  -6.79223733e-02 -2.72924161e-02  4.29888154e-02  4.11278431e-01
  -1.21747881e-01  1.33964358e-04 -4.82635951e-01 -8.23258453e-01
   2.90044843e-03  1.04770930e-02  1.04770930e-02  3.16474061e-01
  -2.91603393e-01 -4.48059053e-04 -4.48059053e-04 -2.11956074e-02
   1.14757576e-02  1.14757576e-02  2.13991992e-01  4.25428791e-01
   9.236045515e-03  1.41535283e-02  1.41535283e-02 -6.99421468e-05
  -7.03892942e-03  2.11925108e-02  5.42289730e-04  1.32431686e-02
   1.32431686e-02  2.52032969e-02  6.70447767e-03 -1.57652797e-02
   3.29509529e-02 -1.76445543e-02 -4.99090504e-04 -5.62255052e-04
  -5.62255052e-04 -9.27821626e-04 -9.27821626e-04  2.52165762e-02
   2.52165762e-02 -6.20699629e-02  4.77016420e-02 -6.73406909e-03
  -4.34379424e-01  4.80174177e-01  2.66107395e-02  2.66107395e-02
   1.06272342e+00 -2.54940299e-01  1.76320769e-02  1.83135442e-02
   2.58548703e-02 -1.71273569e-02  2.77324148e-02  6.47875586e-03
   3.91878129e-02  2.19523807e-02 -7.91176431e-02 -3.95938707e-01
  -5.57587801e-02 -5.57587801e-02 -2.79974557e-02 -3.79393033e-02
   5.09759789e-02 -1.27408458e-04 -1.27408458e-04  5.09244746e-05
   5.09244749e-05  3.14949159e-02 -9.60952552e-03 -9.60952552e-03
   1.21714098e-01 -4.72507300e-02  6.10308126e-04  6.10308126e-04
  -4.65120694e-02 -9.99586537e-02  5.44285772e-03]]\Ridge Regression Intercept: [-1.24172231]

Training: mean absolute error:  0.7025343078227769
Test mean absolute error:  0.7009320623969 89
Training average error rate:  0.5407735980566852
Test average error rate:  0.5394821140280315
Explained variance in training set is:  0.04953417446802133
Explained variance in test set is:  0.04486034715527376
R2 Score training set:  0.04953417446802111
R2 Score testing set:  0.04482933788042309
```

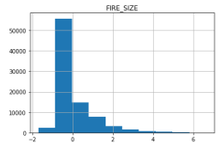




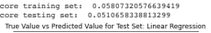
LINEAR REGRESSION and RIDGE REGRESSION for Summer Months

```
# LINEAR REGRESSION FOR SUMMER MONTHS
log_fire_size = pd.DataFrame(np.log(FIRE_SIZE))
log_fire_size = log_fire_size.replace([np.inf, -np.inf], np.nan)
log_fire_size = log_fire_size.fillna(0)
log_fire_size = (log_fire_size - log_fire_size.mean()) / log_fire_size.std()
```

```
log_fire_size.hist()
plt.show()

#x = 11
# x = 6
# x = 1
summer_fires = ((cleaned['DISCOVERY_MONTH'] >= 5) & (cleaned['DISCOVERY_MONTH'] <= 8))
summer_fires_log_size = log_fire_size[summer_fires]
summer_fires_df = fire_size_prediction_df[summer_fires]
print(len(summer_fires_log_size))

run_full_linear_regression_with_accuracy(summer_fires_df, summer_fires_log_size)
run_full_ridge_regression_with_accuracy(summer_fires_df, summer_fires_log_size)
```




```
56358
Response Minimum: FIRE_SIZE   -1.698422
dtype: float64
Response Maximum: FIRE_SIZE    6.638909
dtype: float64
Linear Regression Coefficients: [[ 8.00199778e-02  8.00199778e-02 -3.32844170e-01 -3.32844170e-01
  -7.13481383e-02  2.22907407e-03 -9.08874569e-03 -9.08874569e-03
   2.41337775e-01 -4.47504277e-03 -3.52823668e-02 -3.52823668e-02
   1.45353282e-02  1.45353282e-02  1.39738546e-02  1.39738546e-02
   5.44445904e-01  5.44445904e-01  1.08927412e-01  3.81038848e-01
   3.81038848e-01 -1.36752718e-01 -4.40161386e-03  6.73093842e-02
  -2.26870806e-01 -1.00952004e-02 -1.00952004e-02  2.27601478e-01
   2.27601478e-01  5.76593355e-02  2.68779279e-03  2.68779279e-03
  -6.02754586e-01 -6.02754586e-01 -9.29554666e-03 -9.29554666e-03
  -1.81550739e-01 -1.81550739e-01 -5.06346483e-02 -5.06346483e-02
   2.45690621e-02  2.45690621e-02 -3.61959915e-01 -3.61959915e-01
  -6.29060146e-03 -6.29060146e-03 -2.28366913e-02 -2.28366913e-02
   7.68524820e-02 -3.18684779e-02 -3.18684779e-02  5.67121899e-02
   5.67121899e-02 -2.91289300e-01  3.12459790e+00 -1.37321192e+00
  -5.24557664e-03 -5.24557664e-03  2.53790085e-03  2.53790085e-03
  -1.66451761e+00 -2.33753684e-01  2.36001568e-01  2.36001568e-01
  -9.38464605e-02 -9.38464605e-02  9.63567792e-02  9.63567792e-02
  -6.54385450e-03 -6.54385450e-03  1.03221849e-01  1.35825345e-01
  -1.12471115e+00 -8.31572351e-02  3.00681080e+00 -4.72850404e-02
  -3.80787191e+00 -2.62667441e-02 -2.62667441e-02  3.05782958e+00
  -2.14783711e+00  1.03185987e-02  8.45828131e-02  6.74947759e-01
  -2.01683169e-01  1.61733335e-01  1.61733335e-01  2.72265181e-02
   2.72265181e-02  9.93674248e-02  9.93674248e-02 -4.94310459e-01
  -4.94310459e-01  9.80747250e-04  4.33896321e-02  3.51314336e-02
  -3.26226683e-02 -3.26226683e-02  9.44312293e-02  2.78046425e-01
   1.12136965e+00 -1.01022346e-01 -9.66737591e-02  1.71567424e-01
   4.34134236e-02 -1.59058930e-03 -1.59058930e-03  2.01645017e-03
   2.01645017e-03  8.53294360e-02 -6.53125325e-04 -6.53125325e-04
   3.22860302e-02 -4.39400203e-01  6.79531219e-02  1.90947523e-02
   3.83528112e+00  2.61192681e-01  4.97511585e-02  1.48704168e-01
  -1.27971588e-02  8.52791539e-04 -1.08925136e-01 -1.40549403e-01
  -9.76651765e-03  2.02190868e-02 -8.12967428e-02 -2.68813121e-01
   4.17847747e-02  1.19780167e-01 -2.55622939e-01  1.19358763e+00
   3.27058876e-02  4.86401636e-02 -5.48208852e-02 -7.57334994e-02
  -1.50581851e-02 -1.21176517e-02 -1.21176517e-02  8.31952402e-02
   8.59286537e-02  9.25738891e-02 -5.60675360e-01  8.28369808e-02
  -8.30354309e-02  8.84521005e-02 -8.22844772e-02 -8.22844772e-02
  -9.58010414e-02  4.05485425e-03 -9.11479246e-01  1.39425207e-02
  -8.35245069e-02  1.88402799e-02 -9.15316593e-02  1.17975189e+00
  -3.10962419e-02  2.78581840e-03 -8.58397460e-02 -2.40204385e+00
   6.92812887e-03  4.77215171e-01  4.77215171e-01  9.65149896e-01
   6.73891394e-02  1.99011509e-02  1.99011509e-02 -5.20726746e-02
  -4.57180688e-03 -4.57180688e-03  1.10558751e-01 -4.96261661e-02
   2.48283888e-02  2.40189639e-02  2.40189639e-02 -1.37934644e-02
   5.98361800e-02  4.76831666e-02 -4.66710912e-03  1.90703486e-01
   1.90703486e-01 -2.50928862e-01  7.93213796e-02  1.26799380e-20
  -2.54964884e-02 -1.04667414e-02 -2.27994611e-02  7.04337045e-03
   7.04337045e-03 -7.87998325e-04 -7.87998325e-04  5.42040287e-02
   5.42040287e-02 -7.92926196e-02  4.52663290e-02 -2.48030955e-02
  -4.23430574e-01  5.88052514e-01  8.13661380e-02  8.13661380e-02
   1.01231801e+00 -3.10398508e-01  5.93318447e-03  1.53366817e-02
  -9.64087043e-02 -2.29840247e-02  3.52315157e-02 -5.78219205e-02
   1.28643335e-02  4.79628469e-02 -8.55298644e-02  1.51494773e-01
  -6.74183060e-02 -6.74183060e-02  6.88345893e-03 -1.06329549e-01
   8.36930047e-02 -6.74179186e-02 -6.74179186e-02 -6.12472235e-03
  -6.12472235e-03  2.84361158e-02 -9.69245284e-03 -9.69245284e-03
  -9.40578183e-02 -9.72681025e-02  5.49710874e-02  5.49710874e-02
  -6.73759774e-02 -1.12993610e-01  1.93417947e-02]]
Linear Regression Intercept: [-2.37450912]

Training: mean absolute error:  0.7235662080929671
Test mean absolute error:  0.7328722114890088
Training average error rate:  0.5581052030084253
Test average error rate:  0.5597196652795539
Explained variance in training set is:  0.058073205766394076
Explained variance in test set is:  0.05112624256973641
R2 Score training set:  0.058073205766394119
R2 Score testing set:  0.051065833881329
```

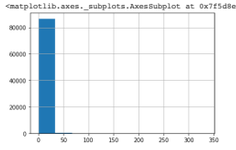


Fire Duration Prediction using Linear Regression and Ridge Regression

Here, we predict the duration of a fire using linear regression

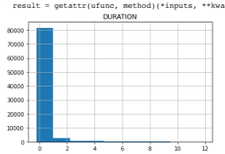
```
FIRE_DURATION = numerical["DURATION"]
FIRE_DURATION.hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5d8e2b97d0>
```




```
log_fire_duration = pd.DataFrame(np.log(FIRE_DURATION))
log_fire_duration = log_fire_duration.replace([np.inf, -np.inf], np.nan)
log_fire_duration = log_fire_duration.fillna(0)
log_fire_duration = (log_fire_duration - log_fire_duration.mean()) / log_fire_duration.std()
log_fire_duration.hist()
plt.show()

# run_full_linear_regression_with_accuracy(fire_duration_prediction_df[:SUBSET], log_fire_duration[:SUBSET])
run_full_ridge_regression_with_accuracy(fire_duration_prediction_df[:SUBSET], log_fire_duration[:SUBSET])

```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/series.py:726: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

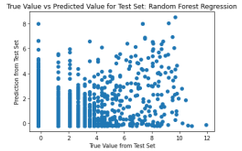



```
Response Minimum: DURATION   -0.214565
dtype: float64
Response Maximum: DURATION    11.92416
dtype: float64
Ridge Regression Coefficients: [[ 1.02358088e-01  1.17706857e-02  2.77658530e-02 -4.89525767e-02
   1.95109615e-02  2.60871570e-02  1.17886907e-03 -3.39702365e-02
  -3.39702365e-02  1.14883355e-02  1.14883355e-02 -1.79400296e-03
  -1.79400295e-03  1.37626317e-03  1.37626317e-03  1.01626380e-01
   1.01626380e-01  1.85498678e-02  1.54935616e-01 -4.37212136e-02
  -3.65462451e-02 -3.65462451e-02 -5.29869105e-02 -5.92115148e-02
  -5.92115149e-02 -7.51207610e-02  2.46944372e-02  2.46944372e-02
   2.76259866e-02  2.95448724e-03 -6.11598490e-02  1.49373314e-02
   1.33683946e-03 -1.20237357e-02  5.05777989e-01 -1.00976037e-01
   2.66687688e-02  2.66687688e-02 -1.06019517e-02 -1.06019517e-02
  -1.72863067e-02 -1.91313449e-02 -2.77578655e-02 -2.77578655e-02
  -2.06476751e-03 -2.06476751e-03 -5.64733621e-02  1.03990442e-02
   1.77977628e-02  2.13905891e-02 -2.90507249e-02  1.11060190e-02
   2.38643160e-02  1.08445871e-02  4.04342440e-02 -3.09167089e-03
  -3.09167088e-03  7.05168060e-03  7.03837743e-03  7.03837743e-03
   6.99616520e-03  6.99616520e-03 -2.45759277e-01  3.09045845e-02
```

```
1 run_full_random_forest_regression_with_accuracy(fire_duration_prediction_df[:SUBSET], log_fire_duration[:SUBSET])
```

```
Response Minimum: DURATION   -0.214565
dtype: float64
Response Maximum: DURATION    11.92416
dtype: float64
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:154: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
Training: mean absolute error:  0.11779824262916068
Test mean absolute error:  0.3229214012623603 5
Training average error rate:  0.009539562228133569
Test average error rate:  0.04341452036099269
Explained variance in training set is:  0.8867181072640216
Explained variance in test set is:  0.22640951091631878
```

True Value vs Predicted Value for Test Set: Random Forest Regression

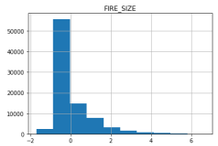


True Value vs Predicted Value for Training Set: Random Forest Regression


```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-30-f16464101162a> in <module>()
----> 1 run_full_random_forest_regression_with_accuracy(fire_duration_prediction_df[:SUBSET], log_fire_duration[:SUBSET])

                                 3 frames
/usr/local/lib/python3.7/dist-packages/pandas/core/ops/__init__.py in to_series(right)
    464             if len(left.columns) != len(right):
    465                 raise ValueError(
--> 466                     msg.format(req_len=len(left.columns), given_len=len(right))
    467                 )
    468             right = left._constructor_sliced(right, index=left.columns)

ValueError: Unable to coerce to Series, length must be 1: given 17421
```

Random Forest Regression FIRE_SIZE

```
1 log_fire_size = pd.DataFrame(np.log(FIRE_SIZE))
2 log_fire_size = log_fire_size.replace([np.inf, -np.inf], np.nan)
3 log_fire_size = log_fire_size.fillna(0)
4 log_fire_size = (log_fire_size - log_fire_size.mean()) / log_fire_size.std()
5 log_fire_size.hist()
6 plt.show()
7
8 # run_full_SVM_regression_with_accuracy(fire_size_prediction_df[:SUBSET], log_fire_size[:SUBSET])
9 run_full_random_forest_regression_with_accuracy(fire_size_prediction_df[:-1], log_fire_size[:-1])
10
```



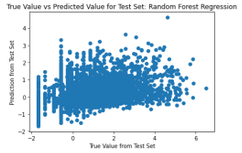

```
Response Minimum: FIRE_SIZE   -1.698422
dtype: float64
Response Maximum: FIRE_SIZE    6.638909
dtype: float64
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:154: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
```

Double-click (or enter) to edit

```
1 # Random forests for summer months
2 run_full_random_forest_regression_with_accuracy(summer_fires_df, summer_fires_log_size)
```

```
Response Minimum: FIRE_SIZE   -1.698422
dtype: float64
Response Maximum: FIRE_SIZE    6.638909
dtype: float64
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:154: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,), for example using ravel().
Training: mean absolute error:  0.25446062326206703
Test mean absolute error:  0.6681863760662583
Training average error rate:  0.17145240208135336
Test average error rate:  0.4603202742169611
Explained variance in training set is:  0.8693374232241285
Explained variance in test set is:  0.14721311827405048
```

True Value vs Predicted Value for Test Set: Random Forest Regression



True Value vs Predicted Value for Training Set: Random Forest Regression


```
1
```

Model 2: Random Forest Classifier

```
# Model 2:
stat = numerical['STAT_CAUSE_CODE']
stat.value_counts()

cleaned_stat1 = numerical[numerical.STAT_CAUSE_CODE != 13.0]
cleaned_stat = cleaned_stat1[cleaned_stat1.STAT_CAUSE_CODE != 9.0]

col_index = cleaned_stat.columns.get_loc('STAT_CAUSE_CODE')
#print(col_index)
cleaned_stat.head()

# use either a random forest or neural network to predict the
```

	Unnamed: 0	BODIE	BROOKS	CONRASSET	CONTAINMENT_MONTH	CONT_DATE	CONT_DOY	CONT_TIME	DISCOVERY_DATE	DISCOVERY_DOY	DISCOVERY_MONTH	DISCOVERY_TIME	EEL_RIVER	FIRE_SIZE	FIRE_YEAR	FOD_ID	HELL_HOLE	HERNANDEZ	HUNTER_MOUNTAIN	JUANITA_LAKE	LADDER_BUTTE	LAS_TABLAS	LA_HONDA	OAK_CREEK	PANAMINT	PILOT_HILL	PRIMARY_STATION_10_MONTHS_PRIOR_EXTREME_MAXIMUM_TEMPERATURE_MONTH	PRIMA
1	1	-1.237103	-0.939629	-0.802799	5	2453137.5	133.0	1530.0	2453137.5	133	5	845.0	-0.597768	0.25	2004	2	-1.471873	-0.562875	-0.243079	-0.487813	-0.669480	-0.281374	-0.869095	-0.451833	-0.057172	-1.280961	1.107121	
2	2	-1.070104	-1.059637	-0.891098	6	2453156.5	152.0	2024.0	2453156.5	152	6	1921.0	-0.695749	0.10	2004	3	-1.410920	-0.543260	-0.109734	-0.533929	-0.714976	-0.262071	-0.950563	-0.317057	0.059122	-1.392229	0.449504	
3	3	-1.588084	-0.752356	-0.586044	7	2453189.5	185.0	1400.0	2453184.5	180	6	1600.0	-0.401895	0.10	2004	4	-1.213089	-0.755069	-0.564467	-0.322165	-0.486222	-0.477673	-0.836997	-0.795000	-0.341413	-1.070171	0.054933	
4	4	-1.578681	-0.759860	-0.590604	7	2453189.5	185.0	1200.0	2453184.5	180	6	1600.0	-0.407106	0.10	2004	5	-1.217813	-0.758604	-0.557924	-0.324379	-0.488626	-0.480494	-0.845502	-0.788869	-0.335760	-1.077584	0.054933	
5	5	-1.475062	-0.828420	-0.650928	7	2453187.5	183.0	1600.0	2453186.5	182	7	1800.0	-0.466098	0.10	2004	6	-1.287362	-0.734812	-0.468792	-0.365956	-0.534467	-0.452476	-0.887998	-0.696533	-0.257419	-1.153409	-0.142352	

5 rows × 246 columns

```
from sklearn.ensemble import RandomForestClassifier
# Instantiate model with 135 decision trees (figured out in the next few blocks of code that this is optimal)
forest = RandomForestClassifier(n_estimators = 135, oob_score = True)
```

```
x_train_mod2 = np.array(cleaned_stat.iloc[:, col_index+1:])
y_mod2 = cleaned_stat.iloc[:, [col_index]]
y_train_mod2 = np.array(y_mod2)

forest.fit(x_train_mod2, y_train_mod2.ravel())
print(forest.score(x_train_mod2, y_train_mod2))
print(forest.oob_score_)
```

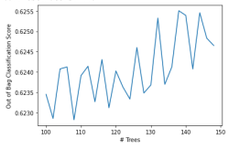
```
1.0
0.6247818786422151
```

```
oob_list = []

for i in range(20, 100, 2):
    print("curr estimators:", i)
    forest = RandomForestClassifier(n_estimators = i, oob_score = True)
    forest.fit(x_train_mod2, y_train_mod2.ravel())
    oob_list.append(forest.oob_score_)
plt.plot(list(range(20, 100, 2)), oob_list)
plt.xlabel('# Trees')
plt.ylabel('Out of Bag Classification Score')
plt.show()
```

```
curr estimators: 20
/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_forest.py:554: UserWarning: Some inputs do not have OOB scores. This probably means too few trees were used to compute any reliable OOB estimates.
  UserWarning,
curr estimators: 22
/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_forest.py:554: UserWarning: Some inputs do not have OOB scores. This probably means too few trees were used to compute any reliable OOB estimates.
  UserWarning,
curr estimators: 24
/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_forest.py:554: UserWarning: Some inputs do not have OOB scores. This probably means too few trees were used to compute any reliable OOB estimates.
  UserWarning,
curr estimators: 26
curr estimators: 28
/usr/local/lib/python3.7/dist-packages/sklearn/ensemble/_forest.py:554: UserWarning: Some inputs do not have OOB scores. This probably means too few trees were used to compute any reliable OOB estimates.
  UserWarning,
curr estimators: 30
curr estimators: 32
curr estimators: 34
curr estimators: 36
```

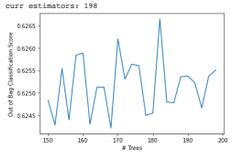
```
oob_list2 = []

for i in range(100, 150, 2):
    print("curr estimators:", i)
    forest = RandomForestClassifier(n_estimators = i, oob_score = True)
    forest.fit(x_train_mod2, y_train_mod2.ravel())
    oob_list2.append(forest.oob_score_)
plt.plot(list(range(100, 150, 2)), oob_list2)
plt.xlabel('# Trees')
plt.ylabel('Out of Bag Classification Score')
plt.show()
```

```
curr estimators: 100
curr estimators: 102
curr estimators: 104
curr estimators: 106
curr estimators: 108
curr estimators: 110
curr estimators: 112
curr estimators: 114
curr estimators: 116
curr estimators: 118
curr estimators: 120
curr estimators: 122
curr estimators: 124
curr estimators: 126
curr estimators: 128
curr estimators: 130
curr estimators: 132
curr estimators: 134
curr estimators: 136
curr estimators: 138
curr estimators: 140
curr estimators: 142
curr estimators: 144
curr estimators: 146
curr estimators: 148
```




```
oob_list3 = []

for i in range(150, 200, 2):
    print("curr estimators:", i)
    forest = RandomForestClassifier(n_estimators = i, oob_score = True)
    forest.fit(x_train_mod2, y_train_mod2.ravel())
    oob_list3.append(forest.oob_score_)
plt.plot(list(range(150, 200, 2)), oob_list3)
plt.xlabel('# Trees')
plt.ylabel('Out of Bag Classification Score')
plt.show()
```

```
curr estimators: 150
curr estimators: 152
curr estimators: 154
curr estimators: 156
curr estimators: 158
curr estimators: 160
curr estimators: 162
curr estimators: 164
curr estimators: 166
curr estimators: 168
curr estimators: 170
curr estimators: 172
curr estimators: 174
curr estimators: 176
curr estimators: 178
curr estimators: 180
curr estimators: 182
curr estimators: 184
curr estimators: 186
curr estimators: 188
curr estimators: 190
curr estimators: 192
curr estimators: 194
curr estimators: 196
curr estimators: 198
```




```
forest = RandomForestClassifier(n_estimators = 182, oob_score = True)
forest.fit(x_train_mod2, y_train_mod2.ravel())
val_list = forest.feature_importances_
idx_list = np.argsort(val_list)[::-1]

print('From high to low:')
for idx in idx_list:
    print('Feature %d: %f' % (idx, val_list[idx]))
```

```
From high to low:
Feature 0: 0.215777
Feature 4: 0.187571
Feature 3: 0.174209
Feature 1: 0.166072
Feature 2: 0.162683
Feature 5: 0.050670
Feature 6: 0.043017
```

```
x_train_mod3 = np.array(cleaned_stat.iloc[:, col_index+1:])
```

```
y_mod3 = cleaned_stat.iloc[:, [col_index]]
y_train_mod3 = np.array(y_mod3)

forest.fit(x_train_mod3, y_train_mod3.ravel())
print(forest.score(x_train_mod3, y_train_mod3))
print(forest.oob_score_)
```

```
1.0
0.6264445395581603
```

```
forest = RandomForestClassifier(n_estimators = 182, oob_score = True)

x_train_mod2 = np.array(cleaned_stat.iloc[:, col_index+1:])
y_mod2 = cleaned_stat.iloc[:, [col_index]]
y_train_mod2 = np.array(y_mod2)

X_train, X_test, y_train, y_test = train_test_split(cleaned_stat, y_train_mod2, test_size=0.2, random_state=2020)

forest.fit(X_train, y_train.ravel())
print(forest.score(X_test, y_test))
print(forest.oob_score_)


```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:446: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
  "X does not have valid feature names, but"
0.8585185185185186
0.8519837023623343
```