```
#https://www.fs.usda.gov/rds/archive/catalog/RDS-2013-0009.6 - DataSet Source
#https://www.kaggle.com/code/emilykchang/stats-project-wildfire-risk/notebook
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import random
import re
from sklearn.neighbors import KNeighborsClassifier
from sklearn import tree, preprocessing
import sklearn.ensemble as ske
from sklearn.model_selection import train_test_split

from plotly.offline import init_notebook_mode, iplot, plot
import plotly as py
init_notebook_mode(connected=True)
import plotly.graph_objs as go

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

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

```
cnx = sqlite3.connect('./drive/MyDrive/Assignments/FPA_FOD_20221014.sqlite')


DF = pd.read_sql_query("SELECT FIRE_YEAR,NWCG_CAUSE_CLASSIFICATION,NWCG_GENERAL_CAUSE,LATITUDE,LONGITUDE,STATE,DISCOVERY_DATE, DISCOVERY_DOY,
print(DF.head())
print(DF.describe())
```

```
   FIRE_YEAR NWCG_CAUSE_CLASSIFICATION  \
0       2005                     Human
1       2004                   Natural
2       2004                     Human
3       2004                   Natural
4       2004                   Natural

                         NWCG_GENERAL_CAUSE   LATITUDE   LONGITUDE STATE  \
0  Power generation/transmission/distribution  40.036944 -121.005833    CA
1                                    Natural  38.933056 -120.404444    CA
2                       Debris and open burning  38.984167 -120.735556    CA
3                                    Natural  38.559167 -119.913333    CA
4                                    Natural  38.559167 -119.933056    CA

  DISCOVERY_DATE  DISCOVERY_DOY  CONT_DATE  CONT_DOY FIRE_SIZE_CLASS  \
0       2/2/2005             33   2/2/2005      33.0               A
1      5/12/2004            133  5/12/2004     133.0               A
2      5/31/2004            152  5/31/2004     152.0               A
3      6/28/2004            180   7/3/2004     185.0               A
4      6/28/2004            180   7/3/2004     185.0               A

   FIRE_SIZE
0       0.10
1       0.25
2       0.10
3       0.10
4       0.10
          FIRE_YEAR      LATITUDE     LONGITUDE  DISCOVERY_DOY      CONT_DOY  \
count  2.303566e+06  2.303566e+06  2.303566e+06   2.303566e+06  1.408753e+06
mean   2.006167e+03  3.696623e+01 -9.635792e+01   1.659714e+02  1.707579e+02
std    8.044361e+00  6.008260e+00  1.664360e+01   8.975278e+01  8.626373e+01
min    1.992000e+03  1.793972e+01 -1.788026e+02   1.000000e+00  1.000000e+00
25%    2.000000e+03  3.301390e+01 -1.110361e+02   9.100000e+01  9.900000e+01
50%    2.006000e+03  3.572250e+01 -9.347009e+01   1.660000e+02  1.760000e+02
75%    2.013000e+03  4.089029e+01 -8.251000e+01   2.310000e+02  2.320000e+02
max    2.020000e+03  7.033060e+01 -6.525694e+01   3.660000e+02  3.660000e+02

          FIRE_SIZE
count  2.303566e+06
mean   7.816088e+01
std    2.630832e+03
min    1.000000e-05
25%    1.000000e-01
50%    8.000000e-01
75%    3.000000e+00
max    6.627000e+05
```

```
DF['NWCG_GENERAL_CAUSE'].value_counts()/len(DF)
```

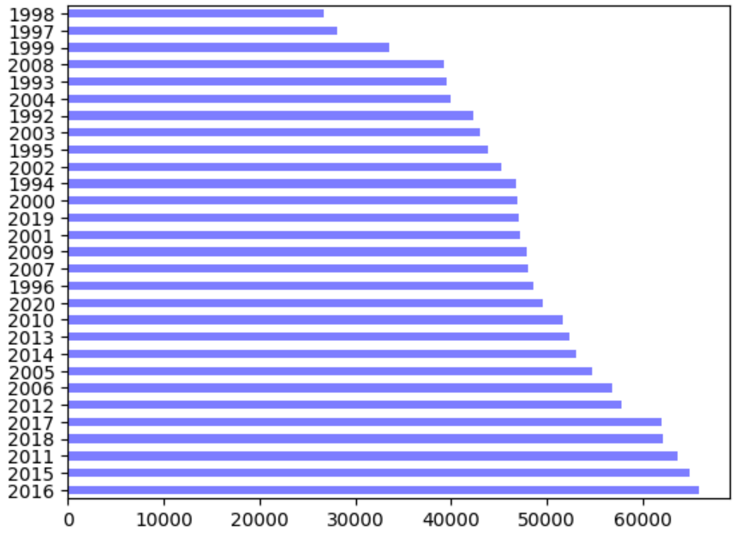
```
Missing data/not specified/undetermined       0.259568
Debris and open burning                       0.232618
Natural                                       0.142092
Arson/incendiarism                            0.139268
Equipment and vehicle use                     0.082619
Recreation and ceremony                       0.043182
Misuse of fire by a minor                     0.028773
Smoking                                       0.027689
Railroad operations and maintenance           0.016189
Power generation/transmission/distribution    0.014175
Fireworks                                     0.008074
Other causes                                  0.004566
Firearms and explosives use                   0.001187
Name: NWCG_GENERAL_CAUSE, dtype: float64
```

```
DAY_TO_CONT=[]
DF=DF.dropna(subset=['CONT_DOY', 'FIRE_SIZE'])
for i in DF.index:
    day2cont=DF.loc[i,'CONT_DOY']-DF.loc[i,'DISCOVERY_DOY']
    DAY_TO_CONT.append(round(day2cont))

DF['DAY_TO_CONT']=DAY_TO_CONT

print(f'data shape\n (observations, features): {DF.shape}\n')
print(DF.head())
```

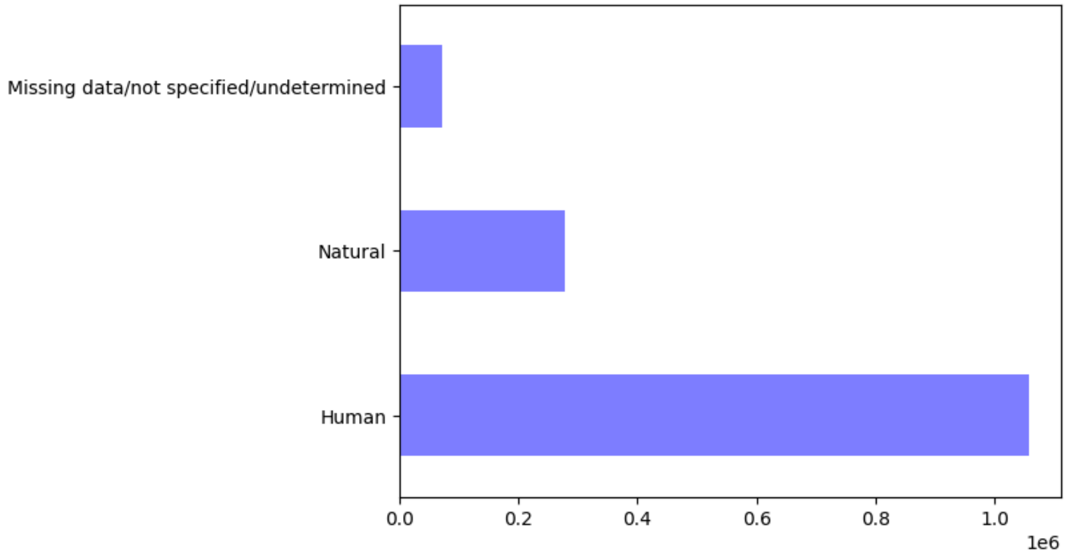
```
data shape
 (observations, features): (1408753, 13)

   FIRE_YEAR NWCG_CAUSE_CLASSIFICATION  \
0       2005                     Human   
1       2004                   Natural   
2       2004                     Human   
3       2004                   Natural   
4       2004                   Natural   

                           NWCG_GENERAL_CAUSE   LATITUDE   LONGITUDE STATE  \
0  Power generation/transmission/distribution  40.036944 -121.005833    CA   
1                                     Natural  38.933056 -120.404444    CA   
2                     Debris and open burning  38.984167 -120.735556    CA   
3                                     Natural  38.559167 -119.913333    CA   
4                                     Natural  38.559167 -119.933056    CA   

  DISCOVERY_DATE  DISCOVERY_DOY  CONT_DATE  CONT_DOY FIRE_SIZE_CLASS  \
0       2/2/2005             33   2/2/2005      33.0               A   
1      5/12/2004            133  5/12/2004     133.0               A   
2      5/31/2004            152  5/31/2004     152.0               A   
3      6/28/2004            180   7/3/2004     185.0               A   
4      6/28/2004            180   7/3/2004     185.0               A   

   FIRE_SIZE  DAY_TO_CONT  
0       0.10            0  
1       0.25            0  
2       0.10            0  
3       0.10            5  
4       0.10            5  
```

```
DF_cols=list(DF.columns)
print(DF_cols)
continuous_features = ['LATITUDE', 'LONGITUDE','DISCOVERY_DATE','DISCOVERY_DOY','CONT_DATE','CONT_DOY','FIRE_SIZE','DATE','DAY_TO_CONT']

DF_cols= [x for x in DF_cols if (x not in continuous_features)]

print(f'count plots for {DF_cols}\n')
for feature in DF_cols:
    print(feature)
    kwargs = dict(alpha=0.5)
    DF[feature].value_counts().plot(kind='barh',**kwargs, color='blue')
    plt.show()
    print("------------")
```

```
['FIRE_YEAR', 'NWCG_CAUSE_CLASSIFICATION', 'NWCG_GENERAL_CAUSE', 'LATITUDE', 'LONGITUDE', 'STATE', 'DISCOVERY_DATE', 'DISCOVERY_DOY',
count plots for ['FIRE_YEAR', 'NWCG_CAUSE_CLASSIFICATION', 'NWCG_GENERAL_CAUSE', 'STATE', 'FIRE_SIZE_CLASS']

FIRE_YEAR
```



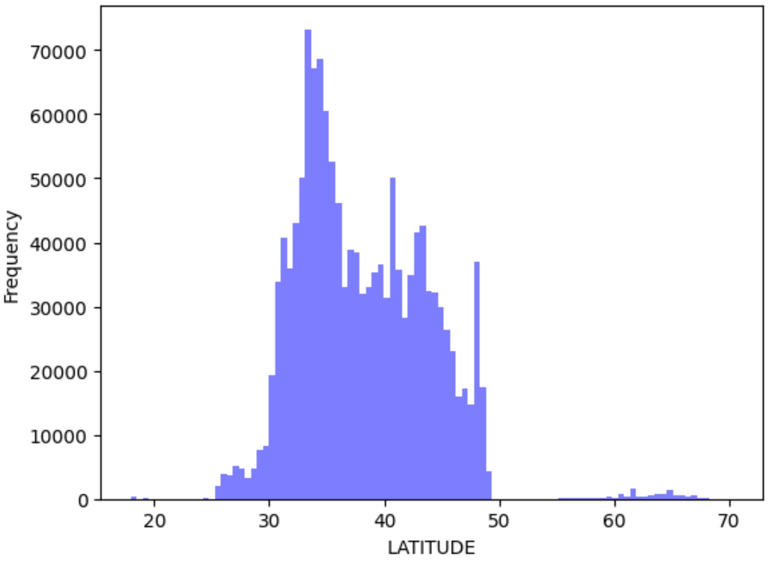
```
------------
NWCG_CAUSE_CLASSIFICATION
```



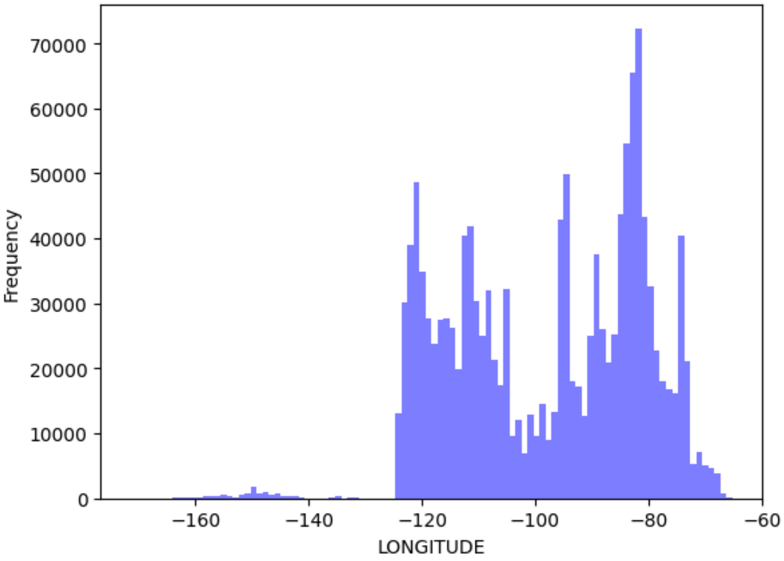
```
------------
NWCG_GENERAL_CAUSE
```

```
DF_cols=list(DF.columns)

DF_col= [x for x in DF_cols if (x in continuous_features and x not in  ['DISCOVERY_DATE','CONT_DATE'])]

print(f'count plots for {DF_col}\n')
for feature in DF_col:
    print(feature)
    kwargs = dict(bins=100,alpha=0.5)
    DF[feature].plot(kind='hist', **kwargs, color="blue")
    plt.xlabel(feature)
    plt.show()
    print("------------")
```

```
count plots for ['LATITUDE', 'LONGITUDE', 'DISCOVERY_DOY', 'CONT_DOY', 'FIRE_SIZE', 'DAY_TO_CONT']

LATITUDE
```



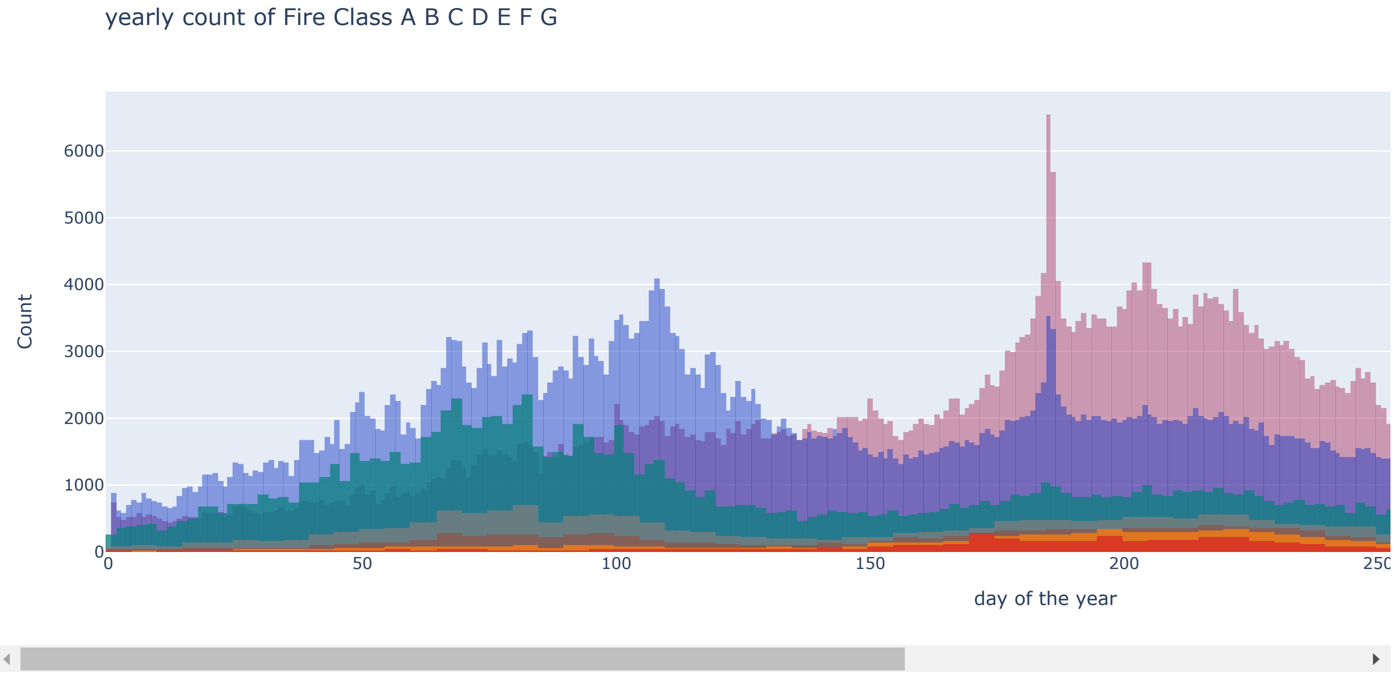
```
------------
LONGITUDE
```



```
------------
DISCOVERY_DOY
```

```
from plotly.offline import iplot
import plotly.graph_objs as go
from plotly.subplots import make_subplots
def enable_plotly_in_cell():
  import IPython
  from plotly.offline import init_notebook_mode
  display(IPython.core.display.HTML('''<script src="/static/components/requirejs/require.js"></script>'''))
  init_notebook_mode(connected=True)
```

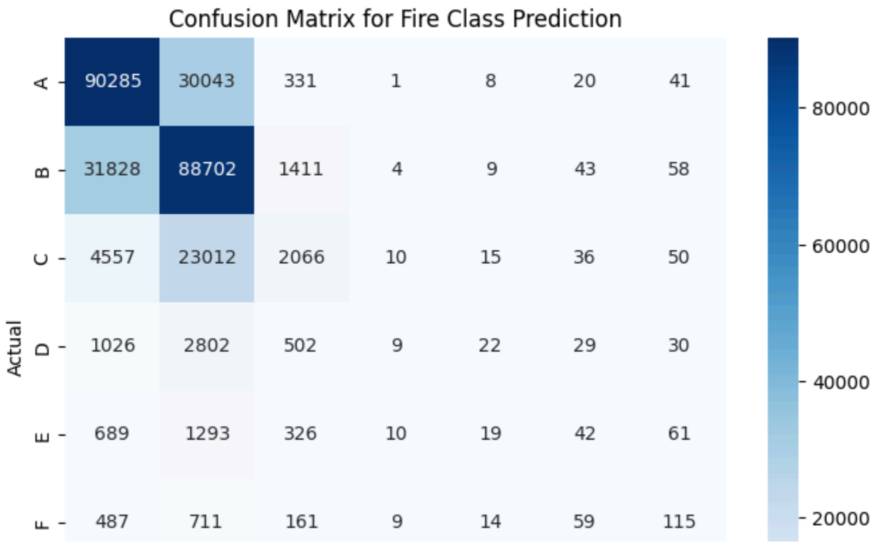
```
DF_A = DF.DISCOVERY_DOY[DF.FIRE_SIZE_CLASS == 'A']
DF_B = DF.DISCOVERY_DOY[DF.FIRE_SIZE_CLASS == 'B']
DF_C = DF.DISCOVERY_DOY[DF.FIRE_SIZE_CLASS == 'C']
DF_D = DF.DISCOVERY_DOY[DF.FIRE_SIZE_CLASS == 'D']
DF_E = DF.DISCOVERY_DOY[DF.FIRE_SIZE_CLASS == 'E']
DF_F = DF.DISCOVERY_DOY[DF.FIRE_SIZE_CLASS == 'F']
DF_G = DF.DISCOVERY_DOY[DF.FIRE_SIZE_CLASS == 'G']

enable_plotly_in_cell()

trace1 = go.Histogram(
    x=DF_A,
    opacity=0.75,
    name = "A",
    marker=dict(color='rgba(171, 50, 96, 0.6)'))

trace2 = go.Histogram(
    x=DF_B,
    opacity=0.75,
    name = "B",
    marker=dict(color='rgba(12, 50, 196, 0.6)'))

trace3 = go.Histogram(
    x=DF_C,
    opacity=0.75,
    name = "C",
    marker=dict(color='rgb(12, 128, 128)'))

trace4 = go.Histogram(
    x=DF_D,
    opacity=0.75,
    name = "D",
    marker=dict(color='rgb(127, 127, 127)'))

trace5 = go.Histogram(
    x=DF_E,
    opacity=0.75,
    name = "E",
    marker=dict(color='rgb(140, 86, 75)'))

trace6 = go.Histogram(
    x=DF_F,
    opacity=0.75,
    name = "F",
    marker=dict(color='rgb(255, 127, 14)'))

trace7 = go.Histogram(
    x=DF_G,
    opacity=0.75,
    name = "G",
    marker=dict(color='rgb(214, 39, 40) '))

data = [trace1,trace2, trace3, trace4, trace5, trace6, trace7]

layout = go.Layout(barmode='overlay',
                   title=' yearly count of Fire Class A B C D E F G',
                   xaxis=dict(title='day of the year'),
                   yaxis=dict( title='Count'),
)

fig = go.Figure(data=data, layout=layout)
iplot(fig)
```



```
# Step 1: Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
import xgboost as xgb
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.utils.class_weight import compute_class_weight

# Step 2: Load your dataset into a pandas DataFrame
# Assuming your DataFrame is named 'df'
df = pd.read_sql_query("SELECT FIRE_YEAR,NWCG_CAUSE_CLASSIFICATION,NWCG_GENERAL_CAUSE,LATITUDE,LONGITUDE,STATE,DISCOVERY_DATE, DISCOVERY_DOY
string_columns = df.select_dtypes(include='object').columns

# Step 3: Preprocess the data
# Drop 'DISCOVERY_DATE' and 'CONT_DATE' columns
df = df.drop(['DISCOVERY_DATE', 'CONT_DATE'], axis=1)
df = df.dropna(subset=['CONT_DOY', 'FIRE_SIZE'])
string_columns = df.select_dtypes(include='object').columns

# Label Encoding for string columns
label_encoder = LabelEncoder()
for col in string_columns:
    df[col] = label_encoder.fit_transform(df[col])

# Step 4: Split the data into training and testing sets
X = df.drop(['FIRE_SIZE_CLASS', 'FIRE_SIZE'], axis=1)
y_fire_class = df[['FIRE_SIZE_CLASS']]
X_train, X_test, y_train_fire_class, y_test_fire_class = train_test_split(X, y_fire_class, test_size=0.2, random_state=42)

# Step 5: Define machine learning model
class_weights = compute_class_weight('balanced', classes=np.unique(y_train_fire_class), y=y_train_fire_class.values.ravel())
model_fire_class = xgb.XGBClassifier(scale_pos_weight=class_weights[1])

# Step 6: Train the model
model_fire_class.fit(X_train, y_train_fire_class.values.ravel())

y_pred_fire_class = model_fire_class.predict(X_test)

accuracy_fire_class = accuracy_score(y_test_fire_class, y_pred_fire_class)

print("Accuracy for fire class prediction:", accuracy_fire_class)
```

```
/usr/local/lib/python3.10/dist-packages/xgboost/core.py:160: UserWarning:

[14:35:01] WARNING: /workspace/src/learner.cc:742:
Parameters: { "scale_pos_weight" } are not used.


Accuracy for fire class prediction: 0.6437492679706549
```

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
conf_mat = confusion_matrix(y_test_fire_class, y_pred_fire_class)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_mat, annot=True, fmt="d", cmap="Blues", xticklabels=label_encoder.classes_, yticklabels=label_encoder.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix for Fire Class Prediction')
plt.show()
```


Confusion Matrix for Fire Class Prediction
Actual
A
B
C
D
E
F