```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import requests
import json
import seaborn as sns
import re
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV, GridSearchCV
import sklearn
%matplotlib inline
import pandas
import numpy
from sklearn.metrics import accuracy_score

pd.set_option('display.max_rows', 10)
pd.set_option('display.max_columns', 10)
def build_dateframe(file):
    """This function will create Pandas DataFrame"""
    df = pd.read_csv(file)
    return df



def draw_histogram(dataframe):
    """create histogram for dataframe passed"""
    dataframe.isna().sum().plot(kind='bar')
    plt.show()
```

```
def type_categorization(df1, df2):
    """create histogram to categorize types of two dataframes"""
    df1['type'].hist(bins=80, alpha=0.8)
    df2['type'].hist(bins=80, alpha=0.8)
    plt.legend(['Disney_Plus', 'Hulu'])
    plt.show()

def missing_values_check(dataframe):
    """This will check missing values within dataframe"""
    movie_missing = dataframe.isnull().sum()
    return movie_missing



def count_entries(file_name, chunk_size, colname):
    """ This will return a dictionary with counts of occurrences as value for key"""
    counts_dict = {}
    for chunk in pd.read_csv(file_name, chunksize=chunk_size):
        for entry in chunk[colname]:
            if entry in counts_dict.keys():
                counts_dict[entry] += 1
            else:
                counts_dict[entry] = 1
    return counts_dict

# loading data from csv file
df_disney_plus= build_dateframe('C:/Users/Swara/FidelityA/UCD Project/disney_plus_titles.csv')
df_hulu = build_dateframe('C:/Users/Swara/FidelityA/UCD Project/hulu_titles.csv')

# Code to demonstrate basic operations on pandas dataframe like find missing & duplicate
```

```
disney_missing = missing_values_check(df_disney_plus)
hulu_missing = missing_values_check(df_hulu)
print('Display Missing Values - Disney Plus: ',  disney_missing)
print('Display Missing Values - Hulu : ', hulu_missing)
disney_dups = df_disney_plus.duplicated(subset=['title'], keep=False)
print(df_disney_plus[disney_dups])
hulu_dups = df_hulu.duplicated(subset=['title'], keep=False)
print(df_hulu[hulu_dups])
# Merging dataframes and removing duplicates

df_merge = pd.concat([df_disney_plus, df_hulu])
print(df_merge.info())
duplicates_combined = df_merge.duplicated(subset=['title'], keep=False)
print(df_merge[duplicates_combined].info())
#Drop duplicates
distinct_movies = df_merge.drop_duplicates(subset=['title','director','cast'])
print(distinct_movies.info())

# Code to demonstrate creating function, dictionary, using iterators

result_counts = count_entries('C:/Users/Swara/FidelityA/UCD Project/disney_plus_titles.csv', 10,
'release_year')
print(result_counts)
# histogram for two dataframe

draw_histogram(df_disney_plus)
draw_histogram(df_hulu)

# Compare type of each dat set(TV,Movie Show)

type_categorization(df_disney_plus, df_hulu)
```

```
# Code to demonstrate loading data from API
response_API = requests.get('https://api.covid19india.org/state_district_wise.json')
#print(response_API.status_code)
data = response_API.text
parse_json = json.loads(data)
active_case = parse_json['Andaman and Nicobar Islands']['districtData']['South Andaman']['active']
print("Active cases in South Andaman:", active_case)

df_tweets = build_dateframe('C:/Users/Swara/FidelityA/UCD Project/Tweets.csv')
hash_tags = []
regex = r"#\b\w\w+\b"
for i in df_tweets['text']:
   word = re.findall(regex, i)
   if len(word) > 0:
      hash_tags.append(word)
print(hash_tags)

# Code to demonstrate regular expression in python, Extracting all hash tags from Tweets
Tweet= pandas.read_csv("C:/Users/Swara/FidelityA/UCD Project/Tweets.csv")
Tweet.head()

(len(Tweet)-Tweet.count())/len(Tweet)
del Tweet['tweet_coord']
del Tweet['airline_sentiment_gold']
del Tweet['negativereason_gold']
Mood_count=Tweet['airline_sentiment'].value_counts()
Index = [1,2,3]
plt.bar(Index,Mood_count)
plt.xticks(Index,['negative','neutral','positive'],rotation=45)
plt.ylabel('Mood Count')
plt.xlabel('Mood')
```

```
plt.title('Count of Moods')
#Machine Learnig Code snippet
#Load Data
df = pd.read_csv("C:/Users/Swara/FidelityA/UCD Project/heart.csv")
df.shape
df.head()
train_df.prognosis.value_counts().head()
#how many class of one feature or target
df["target"].value_counts()
#bar chart
df["target"].value_counts().plot(kind='bar', color=["salmon","lightblue"])
df.info()
#check missing values of all features
df.isna().sum()
df.describe()
#Heart Disease Frequency according to Sex
df.sex.value_counts()
#Compare target and sex column
pd.crosstab(df.target, df.sex)

#Create plot of crosstab
pd.crosstab(df.target, df.sex).plot(kind="bar",figsize=(10,6),color=["salmon","lightblue"])
plt.title("Heart Disease Frerquency for Sex")
plt.xlabel("0 = No Disease, 1=Disease")
plt.ylabel("Amount")
plt.legend(["Female","Male"]);
plt.xticks(rotation=0);
#Create new figure
plt.figure(figsize=(10,6))

#Scatter with positive examples
```

```
plt.scatter(df.age[df.target==1],
        df.thalach[df.target==1],
        c="salmon")

#Scatter with negative examples
plt.scatter(df.age[df.target==0],
        df.thalach[df.target==0],
        c="lightblue");

#Add some helpful info
plt.title("Heart Disease in function of Age and Max Heart Rate")
plt.xlabel("Age")
plt.ylabel("Max Heart Rate")
plt.legend(["Disease","No Disease"])
#Check the distribution of the age column with a histogram
#May check for outliers of the data
df.age.plot.hist()
#Create a hyperparameter grid for LR
log_reg_grid = {"C": np.logspace(-4,4,20),
            "solver": ["liblinear"]}

#Create a hyperparameter grid for RF
rf_grid = {"n_estimators": np.arange(10,1000,50),
        "max_depth": [None,3,5,10],
        "min_samples_split": np.arange(2,20,2),
        "min_samples_leaf": np.arange(1,20,2)}
np.random.seed(42)

# Setup random hyperparameter search for LogisticRegression
rs_log_reg = RandomizedSearchCV(LogisticRegression(),
                        param_distributions=log_reg_grid,
```

```
                    cv=5,
                    n_iter=20,
                    verbose=True)

# Fit random hyperparameter search model for LogisticRegression
rs_log_reg.fit(X_train, y_train)
print('Best Params :- ', end='')
```