Title: EDA for DA Project
Author: Tanay Singh, PES2UG20CS364
Date: 08/09/2022

Importing Libraries

In [1]:
```
import seaborn as sns
import pandas as pd
import numpy as np
import math
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import re
pd.set_option('display.max_columns', None)
pd.set_option('display.expand_frame_repr', False)
pd.set_option('max_colwidth', None)
pd.set_option('display.max_rows', 8)  # 8 is required for .describe()

%load_ext autoreload
%autoreload 2
%matplotlib inline
```

Reading the dataset

In [2]:
```
df = pd.read_csv('KOI_Dataset_1.csv',infer_datetime_format=True)
df.set_index('kepoi_name',inplace=True,drop=False)
df.insert( df.columns.get_loc('kepoi_name'), 'kepoi_star', value=None)
df['kepoi_star'] = df['kepoi_name'].apply(lambda str: re.sub(r'\..*$', '', str))
df.head(5)
```

Out[2]:

	kepid	kepoi_star	kepoi_name	kepler_name	koi_disposition	koi_vet_stat	koi_vet_date	koi_pdisposi
kepoi_name								
K00752.01	10797460	K00752	K00752.01	Kepler-227 b	CONFIRMED	Done	16-08-18	CANDIE
K00752.02	10797460	K00752	K00752.02	Kepler-227 c	CONFIRMED	Done	16-08-18	CANDIE
K00753.01	10811496	K00753	K00753.01	NaN	CANDIDATE	Done	16-08-18	CANDIE
K00754.01	10848459	K00754	K00754.01	NaN	FALSE POSITIVE	Done	16-08-18	FALSE POSI
K00755.01	10854555	K00755	K00755.01	Kepler-664 b	CONFIRMED	Done	16-08-18	CANDIE

Filling NaN Values

In [3]:

```
koi_fillna = {
    # 'kepler_name': '',
    'koi_score':   0,
    'koi_comment': '',
    'koi_tce_delivname': '',
}

for key, value in koi_fillna.items():
    df[key].fillna( value, inplace=True )
```

Number of Rows and Atrribtues in the dataset

In [4]:

```
#Number of rows and attributes in the dataset
nrows = len(df.axes[0])
ncols = len(df.axes[1])

print("Number of rows : ",nrows)
print("Number of columns : ",ncols)
```

```
Number of rows :  9564
Number of columns :  83
```

Total number of missing data in the dataset

In [5]:

```
#Null values
print("Total Number of missing data : ",df.isna().sum().sum())
```

```
Total Number of missing data :  79526
```

Calculating total number of outliers in the dataset

In [6]:

```
numeric_df = df.select_dtypes(include=['float64','int64'])
cols = numeric_df.columns.values
outliers = 0
for col in cols:
    Q1 = (np.percentile(numeric_df[col],25))
    Q3 = (np.percentile(numeric_df[col],75))
    IQR = (Q3 - Q1)

    if(not(math.isnan(IQR))):
        lower_bound = Q1-(1.5*IQR)
        upper_bound = Q3+(1.5*IQR)
        for val in numeric_df[col]:
            if(val<lower_bound or val>upper_bound):
                outliers+=1

print("Total outliers in the dataset : ",outliers)
```

```
Total outliers in the dataset :  13610
```

Checking for any duplicate data

In [7]:

```
duplicate_values = df[df.duplicated()]
print("Duplicate Rows : ")
duplicate_values
```

```
Duplicate Rows :
```

Out[7]:

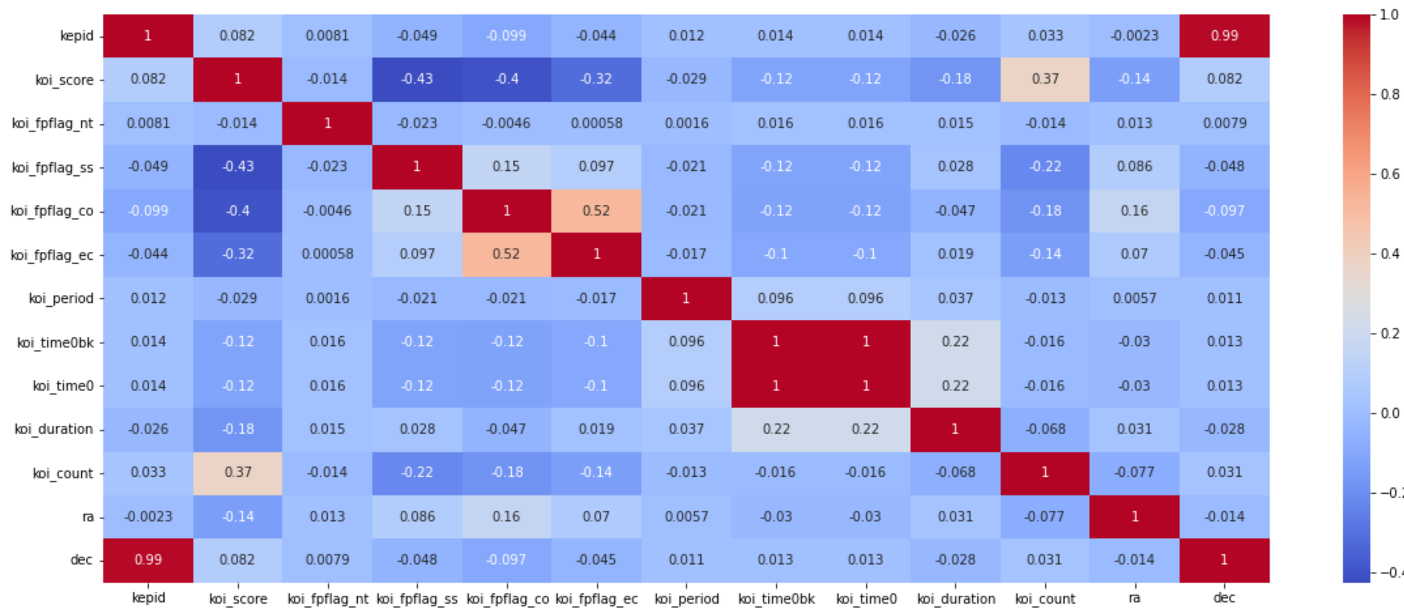
	kepid	kepoi_star	kepoi_name	kepler_name	koi_disposition	koi_vet_stat	koi_vet_date	koi_pdispositior
kepoi_name								

Correlation analysis using Seaborn and Pandas

Here we modify the dataset by dropping all the na values in the columns as these values would have not have contributed to the correlations in any way. After cleaning up the dataset, we plot the correlation plot :

In [8]:

```
plt.figure(figsize=(20,8))
tempdf = df.dropna(axis=1)
corr_matrix = tempdf.corr()
sns.heatmap(corr_matrix,cmap='coolwarm',annot=True)
plt.show()
#corr_matrix.style.background_gradient(cmap='coolwarm')
```



As seen from the heatmap of the correlations across the dataset the following inferences can be made :

- Majority of the data in the dataset is negatively correlated.
- The following show the most amounts of correlation (either +ve or -ve):
  - koi_score with koi_fpflag_ss => correlation value of -0.43
  - koi_score with koi_fpflag_co => correlation value of -0.4
  - koi_score with koi_fpflag_ev => correlation value of -0.32
  - koi_score with koi_count => correlation value of 0.37
  - koi_fpflag_co with koi_kpflac_ec => correlation value of 0.52

Performing Range Transformation

First we find out all the numeric columns presesnt in our dataset

In [9]:

```
numeric_df = df.select_dtypes(include=np.number)
numeric_df.describe()
```

Out[9]:

	kepid	koi_score	koi_fpflag_nt	koi_fpflag_ss	koi_fpflag_co	koi_fpflag_ec	koi_period	koi_time0l
count	9.564000e+03	9564.000000	9564.000000	9564.000000	9564.000000	9564.000000	9564.000000	9564.0000
mean	7.690628e+06	0.404914	0.208595	0.232748	0.197512	0.120033	75.671358	166.1832
std	2.653459e+06	0.471473	4.767290	0.422605	0.398142	0.325018	1334.744046	67.9189
min	7.574500e+05	0.000000	0.000000	0.000000	0.000000	0.000000	0.241843	120.5159
25%	5.556034e+06	0.000000	0.000000	0.000000	0.000000	0.000000	2.733684	132.7617
50%	7.906892e+06	0.000000	0.000000	0.000000	0.000000	0.000000	9.752831	137.2245
75%	9.873066e+06	0.995000	0.000000	0.000000	0.000000	0.000000	40.715178	170.6946
max	1.293514e+07	1.000000	465.000000	1.000000	1.000000	1.000000	129995.778400	1472.5223

Next we check for those columns who have a very wide range of values in them

In [10]:

```
filter = (numeric_df.describe().T['max']>1) | (numeric_df.describe().T['min']<0)
colNames = numeric_df.loc[:,filter].columns.values
```

In [11]:

```
colNames[1:]
```

Out[11]:

```
array(['koi_fpflag_nt', 'koi_period', 'koi_time0bk', 'koi_time0',
       'koi_impact', 'koi_duration', 'koi_depth', 'koi_ror', 'koi_srho',
       'koi_prad', 'koi_sma', 'koi_incl', 'koi_teq', 'koi_insol',
       'koi_dor', 'koi_ldm_coeff2', 'koi_max_sngle_ev', 'koi_max_mult_ev',
       'koi_model_snr', 'koi_count', 'koi_num_transits',
       'koi_tce_plnt_num', 'koi_quarters', 'koi_bin_oedp_sig',
       'koi_steff', 'koi_slogg', 'koi_smet', 'koi_srad', 'koi_smass',
       'ra', 'dec', 'koi_kepmag', 'koi_gmag', 'koi_rmag', 'koi_imag',
       'koi_zmag', 'koi_jmag', 'koi_hmag', 'koi_kmag', 'koi_fwm_sra',
       'koi_fwm_sdec', 'koi_fwm_srao', 'koi_fwm_sdeco', 'koi_fwm_prao',
       'koi_fwm_pdeco', 'koi_dicco_mra', 'koi_dicco_mdec',
       'koi_dicco_msky', 'koi_dikco_mra', 'koi_dikco_mdec',
       'koi_dikco_msky'], dtype=object)
```

Finally we perform MinMaxScaling on the columns extracted in the above step.

Note : We skip the kepid column since it's just an identifier column and will not be useful for future analysis

In [12]:

```
features = colNames[1:]
scaler = MinMaxScaler()
feature_transform = scaler.fit_transform(df[features])
feature_transform= pd.DataFrame(columns=features, data=feature_transform, index=df.index)
feature_transform.head()
```

Out[12]:

	koi_fpflag_nt	koi_period	koi_time0bk	koi_time0	koi_impact	koi_duration	koi_depth	koi_ror	koi_srl
kepoi_name									
K00752.01	0.0	0.000071	0.036999	0.036999	0.001448	0.020980	0.000400	0.000211	0.0032
K00752.02	0.0	0.000417	0.031063	0.031063	0.005813	0.032169	0.000568	0.000267	0.0030

kepoi_name	koi_fpflag_nt	koi_period	koi_time0bk	koi_time0	koi_impact	koi_duration	koi_depth	koi_ror	koi_srl
K00753.01	0.0	0.000151	0.040928	0.040927	0.009613	0.012494	0.007013	0.001530	0.0074
K00754.01	0.0	0.000012	0.036828	0.036828	0.012658	0.017001	0.005247	0.003866	0.0002
K00755.01	0.0	0.000018	0.037781	0.037781	0.006954	0.011571	0.000392	0.000228	0.0020

Pie Chart to check Disposition Counts

Disposition in the dataset stands for the different categories of this KOI from the entire Exoplanet Archive. The current values assigned by NASA for this are :

- CANDIDATE
- FALSE-POSITIVE
- CONFIRMED.

These categories essentially mention if the respective star systems have potential Earth-like planets in them.

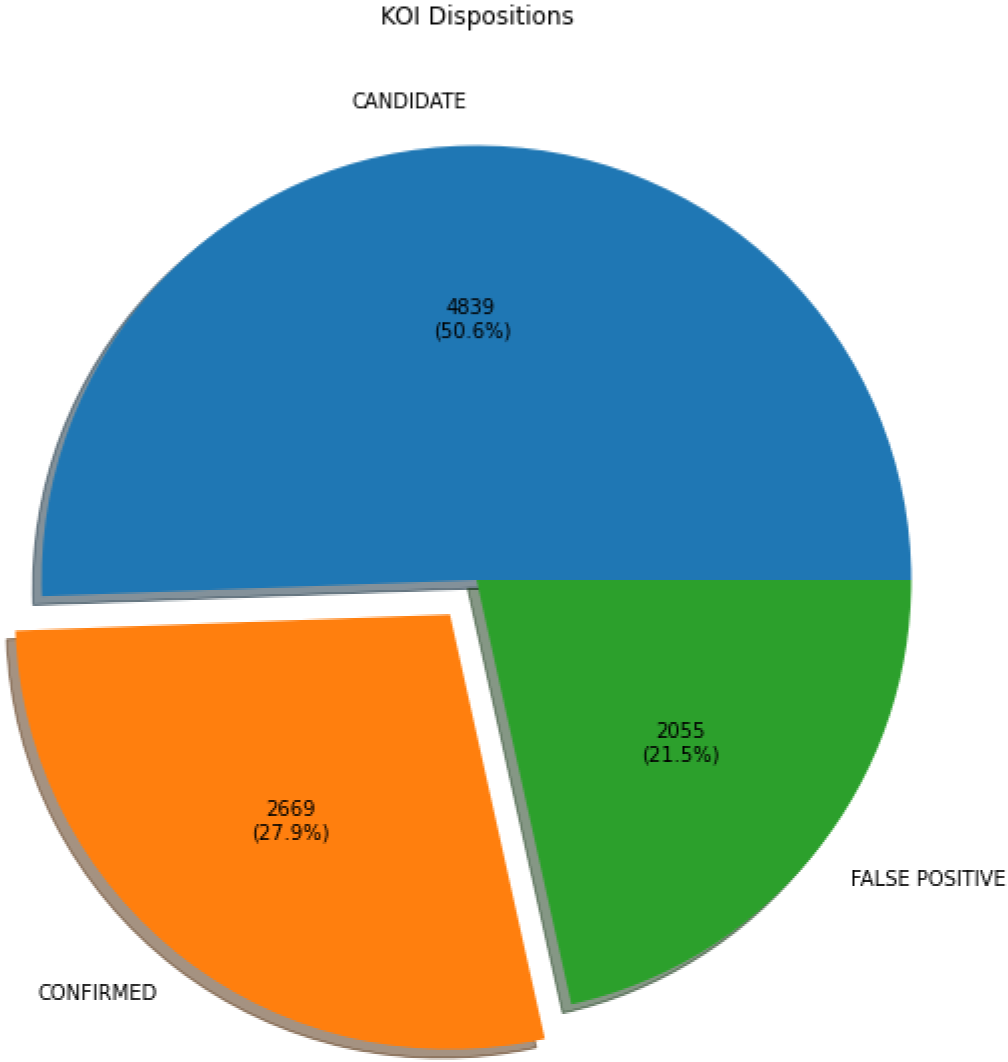
In [13]:

```
koi_disposition_counts = (
    pd.DataFrame(
        df
            .groupby('koi_disposition')
            .size()
            .map(lambda count: {
                "count":      count,
                "percentage": round(count/df.shape[0] * 100, 1)
            })
            .to_dict()
    ).transpose()
)
koi_disposition_counts

def pie_label(pct, allvals):
    absolute = int(pct/100.*np.sum(allvals))
    return "{:d}\n({:.1f}%)".format(absolute, pct)

fig = plt.figure(figsize=(20,10))
fig.patch.set_facecolor('white')
plt.pie(df['koi_disposition'].value_counts(),labels=koi_disposition_counts.index,explode=
plt.title("KOI Dispositions")
plt.show()

koi_disposition_counts['count']
```



Out[13]:
```
CANDIDATE         2056.0
CONFIRMED         2669.0
FALSE POSITIVE    4839.0
Name: count, dtype: float64
```

In [ ]: