Graduate-Project-Adrien-Sade

December 8, 2022

1 DATA C200 - Graduate Project

1.1 Space Exploration

1.1.1 Adrien Sadé

[ ]:
```
try:
  from google.colab import drive
  drive.mount('/content/gdrive', force_remount=True)

  FOLDERNAME = 'DATA\ C200\ -\ Graduate Project'
  %cd /content/gdrive/My\ Drive/$FOLDERNAME
except ImportError:
  pass
```

The chosen dataset to study with is Dataset A from Topic 3: Emerging Researches and Technologies. It is given by NASA, and we'll focus here only on the 'kepler_exoplanet_search.csv' table (even though in my drafts I tried to use the other ones provided, but without capital gain).

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
```

[ ]:
```
pd.options.display.max_columns = None
pd.options.display.max_rows = 30
```

[ ]:
```
kes = pd.read_csv('data/kepler_exoplanet_search.csv', skiprows=143, header=1)
kes.head()
```

[ ]:
```
      kepid kepoi_name   kepler_name koi_disposition koi_vet_stat  \
0  11446443  K00001.01    Kepler-1 b       CONFIRMED         Done
1  10666592  K00002.01    Kepler-2 b       CONFIRMED         Done
```

```
2  10748390  K00003.01     Kepler-3 b       CONFIRMED         Done
3   3861595  K00004.01  Kepler-1658 b       CONFIRMED         Done
4   8554498  K00005.01            NaN       CANDIDATE         Done

  koi_vet_date koi_pdisposition  koi_score  koi_fpflag_nt  koi_fpflag_ss  \
0   2018-08-16        CANDIDATE        NaN              0              0
1   2018-08-16        CANDIDATE        NaN              0              1
2   2018-08-16        CANDIDATE        NaN              0              0
3   2018-08-16        CANDIDATE        NaN              0              0
4   2018-08-16        CANDIDATE        NaN              0              0

   koi_fpflag_co  koi_fpflag_ec        koi_disp_prov  koi_comment  koi_period  \
0              0              0  q1_q17_dr25_sup_koi          NaN    2.470613
1              0              0  q1_q17_dr25_sup_koi          NaN    2.204735
2              0              0  q1_q17_dr25_sup_koi          NaN    4.887803
3              0              0  q1_q17_dr25_sup_koi          NaN    3.849372
4              0              0  q1_q17_dr25_sup_koi          NaN    4.780328

   koi_period_err1  koi_period_err2  koi_time0bk  koi_time0bk_err1  \
0     2.700000e-08    -2.700000e-08   122.763305          0.000009
1     4.300000e-08    -4.300000e-08   121.358542          0.000016
2     4.660000e-07    -4.660000e-07   124.813081          0.000075
3     2.344000e-06    -2.344000e-06   157.526686          0.000488
4     8.520000e-07    -8.520000e-07   132.974086          0.000148

   koi_time0bk_err2    koi_time0  koi_time0_err1  koi_time0_err2  koi_eccen  \
0         -0.000009  2454955.763        0.000009       -0.000009        NaN
1         -0.000016  2454954.359        0.000016       -0.000016        NaN
2         -0.000075  2454957.813        0.000075       -0.000075        NaN
3         -0.000488  2454990.527        0.000488       -0.000488        NaN
4         -0.000148  2454965.974        0.000148       -0.000148        NaN

   koi_eccen_err1  koi_eccen_err2  koi_longp  koi_longp_err1  koi_longp_err2  \
0             NaN             NaN        NaN             NaN             NaN
1             NaN             NaN        NaN             NaN             NaN
2             NaN             NaN        NaN             NaN             NaN
3             NaN             NaN        NaN             NaN             NaN
4             NaN             NaN        NaN             NaN             NaN

   koi_impact  koi_impact_err1  koi_impact_err2  koi_duration  \
0         NaN              NaN              NaN           NaN
1         NaN              NaN              NaN           NaN
2         NaN              NaN              NaN           NaN
3         NaN              NaN              NaN           NaN
4         NaN              NaN              NaN           NaN

   koi_duration_err1  koi_duration_err2  koi_ingress  koi_ingress_err1  \
```

```
0                NaN                NaN        NaN              NaN
1                NaN                NaN        NaN              NaN
2                NaN                NaN        NaN              NaN
3                NaN                NaN        NaN              NaN
4                NaN                NaN        NaN              NaN

   koi_ingress_err2  koi_depth  koi_depth_err1  koi_depth_err2  koi_ror  \
0               NaN        NaN             NaN             NaN      NaN
1               NaN        NaN             NaN             NaN      NaN
2               NaN        NaN             NaN             NaN      NaN
3               NaN        NaN             NaN             NaN      NaN
4               NaN        NaN             NaN             NaN      NaN

   koi_ror_err1  koi_ror_err2  koi_srho  koi_srho_err1  koi_srho_err2  \
0           NaN           NaN       NaN            NaN            NaN
1           NaN           NaN       NaN            NaN            NaN
2           NaN           NaN       NaN            NaN            NaN
3           NaN           NaN       NaN            NaN            NaN
4           NaN           NaN       NaN            NaN            NaN

   koi_fittype  koi_prad  koi_prad_err1  koi_prad_err2  koi_sma  koi_sma_err1  \
0          NaN       NaN            NaN            NaN      NaN           NaN
1          NaN       NaN            NaN            NaN      NaN           NaN
2          NaN       NaN            NaN            NaN      NaN           NaN
3          NaN       NaN            NaN            NaN      NaN           NaN
4          NaN       NaN            NaN            NaN      NaN           NaN

   koi_sma_err2  koi_incl  koi_incl_err1  koi_incl_err2  koi_teq  \
0           NaN       NaN            NaN            NaN      NaN
1           NaN       NaN            NaN            NaN      NaN
2           NaN       NaN            NaN            NaN      NaN
3           NaN       NaN            NaN            NaN      NaN
4           NaN       NaN            NaN            NaN      NaN

   koi_teq_err1  koi_teq_err2  koi_insol  koi_insol_err1  koi_insol_err2  \
0           NaN           NaN        NaN             NaN             NaN
1           NaN           NaN        NaN             NaN             NaN
2           NaN           NaN        NaN             NaN             NaN
3           NaN           NaN        NaN             NaN             NaN
4           NaN           NaN        NaN             NaN             NaN

   koi_dor  koi_dor_err1  koi_dor_err2  koi_limbdark_mod  koi_ldm_coeff4  \
0      NaN           NaN           NaN               NaN             NaN
1      NaN           NaN           NaN               NaN             NaN
2      NaN           NaN           NaN               NaN             NaN
3      NaN           NaN           NaN               NaN             NaN
4      NaN           NaN           NaN               NaN             NaN
```

```
   koi_ldm_coeff3  koi_ldm_coeff2  koi_ldm_coeff1       koi_parm_prov  \
0             NaN             NaN             NaN  q1_q17_dr25_sup_koi
1             NaN             NaN             NaN  q1_q17_dr25_sup_koi
2             NaN             NaN             NaN  q1_q17_dr25_sup_koi
3             NaN             NaN             NaN  q1_q17_dr25_sup_koi
4             NaN             NaN             NaN  q1_q17_dr25_sup_koi

   koi_max_sngle_ev  koi_max_mult_ev  koi_model_snr  koi_count  \
0               NaN              NaN            NaN          1
1               NaN              NaN            NaN          1
2               NaN              NaN            NaN          1
3               NaN              NaN            NaN          1
4               NaN              NaN            NaN          2

   koi_num_transits  koi_tce_plnt_num koi_tce_delivname  koi_quarters  \
0               NaN               NaN   q1_q17_dr25_tce           NaN
1               NaN               NaN   q1_q17_dr25_tce           NaN
2               NaN               NaN   q1_q17_dr25_tce           NaN
3               NaN               NaN   q1_q17_dr25_tce           NaN
4               NaN               NaN   q1_q17_dr25_tce           NaN

   koi_bin_oedp_sig  koi_trans_mod  koi_model_dof  koi_model_chisq  \
0               NaN            NaN            NaN              NaN
1               NaN            NaN            NaN              NaN
2               NaN            NaN            NaN              NaN
3               NaN            NaN            NaN              NaN
4               NaN            NaN            NaN              NaN

                                    koi_datalink_dvr  koi_datalink_dvs  \
0  011/011446/011446443/dv/kplr011446443-20160209…               NaN
1  010/010666/010666592/dv/kplr010666592-20160209…               NaN
2  010/010748/010748390/dv/kplr010748390-20160209…               NaN
3  003/003861/003861595/dv/kplr003861595-20160209…               NaN
4  008/008554/008554498/dv/kplr008554498-20160209…               NaN

   koi_steff  koi_steff_err1  koi_steff_err2  koi_slogg  koi_slogg_err1  \
0        NaN             NaN             NaN        NaN             NaN
1        NaN             NaN             NaN        NaN             NaN
2        NaN             NaN             NaN        NaN             NaN
3        NaN             NaN             NaN        NaN             NaN
4        NaN             NaN             NaN        NaN             NaN

   koi_slogg_err2  koi_smet  koi_smet_err1  koi_smet_err2  koi_srad  \
0             NaN       NaN            NaN            NaN       NaN
1             NaN       NaN            NaN            NaN       NaN
2             NaN       NaN            NaN            NaN       NaN
```

```
3            NaN       NaN            NaN            NaN       NaN
4            NaN       NaN            NaN            NaN       NaN

   koi_srad_err1  koi_srad_err2  koi_smass  koi_smass_err1  koi_smass_err2  \
0            NaN            NaN        NaN             NaN             NaN
1            NaN            NaN        NaN             NaN             NaN
2            NaN            NaN        NaN             NaN             NaN
3            NaN            NaN        NaN             NaN             NaN
4            NaN            NaN        NaN             NaN             NaN

   koi_sage  koi_sage_err1  koi_sage_err2  koi_sparprov         ra        dec  \
0       NaN            NaN            NaN           NaN  286.80847  49.316399
1       NaN            NaN            NaN           NaN  292.24728  47.969521
2       NaN            NaN            NaN           NaN  297.70935  48.080853
3       NaN            NaN            NaN           NaN  294.35654  38.947380
4       NaN            NaN            NaN           NaN  289.73972  44.647419

   koi_kepmag  koi_gmag  koi_rmag  koi_imag  koi_zmag  koi_jmag  koi_hmag  \
0      11.338    11.736    11.275    11.168    11.126    10.232     9.920
1      10.463    10.935    10.490       NaN       NaN     9.555     9.344
2       9.174    10.665     9.479       NaN       NaN     7.608     7.131
3      11.432    11.755    11.354    11.294    11.305    10.475    10.266
4      11.665    12.085    11.601    11.485    11.458    10.542    10.257

   koi_kmag  koi_fwm_stat_sig  koi_fwm_sra  koi_fwm_sra_err  koi_fwm_sdec  \
0     9.846               NaN          NaN              NaN           NaN
1     9.334               NaN          NaN              NaN           NaN
2     7.009               NaN          NaN              NaN           NaN
3    10.195               NaN          NaN              NaN           NaN
4    10.213               NaN          NaN              NaN           NaN

   koi_fwm_sdec_err  koi_fwm_srao  koi_fwm_srao_err  koi_fwm_sdeco  \
0               NaN           NaN               NaN            NaN
1               NaN           NaN               NaN            NaN
2               NaN           NaN               NaN            NaN
3               NaN           NaN               NaN            NaN
4               NaN           NaN               NaN            NaN

   koi_fwm_sdeco_err  koi_fwm_prao  koi_fwm_prao_err  koi_fwm_pdeco  \
0                NaN           NaN               NaN            NaN
1                NaN           NaN               NaN            NaN
2                NaN           NaN               NaN            NaN
3                NaN           NaN               NaN            NaN
4                NaN           NaN               NaN            NaN

   koi_fwm_pdeco_err  koi_dicco_mra  koi_dicco_mra_err  koi_dicco_mdec  \
0                NaN            NaN                NaN             NaN
```

```
1                    NaN            NaN                 NaN              NaN
2                    NaN            NaN                 NaN              NaN
3                    NaN            NaN                 NaN              NaN
4                    NaN            NaN                 NaN              NaN

   koi_dicco_mdec_err  koi_dicco_msky  koi_dicco_msky_err  koi_dikco_mra  \
0                 NaN             NaN                 NaN            NaN
1                 NaN             NaN                 NaN            NaN
2                 NaN             NaN                 NaN            NaN
3                 NaN             NaN                 NaN            NaN
4                 NaN             NaN                 NaN            NaN

   koi_dikco_mra_err  koi_dikco_mdec  koi_dikco_mdec_err  koi_dikco_msky  \
0                NaN             NaN                 NaN             NaN
1                NaN             NaN                 NaN             NaN
2                NaN             NaN                 NaN             NaN
3                NaN             NaN                 NaN             NaN
4                NaN             NaN                 NaN             NaN

   koi_dikco_msky_err
0                 NaN
1                 NaN
2                 NaN
3                 NaN
4                 NaN
```

The dataset is huge, both in terms of examples and features. Let's dig into that, clean the data and pre-process it, and focus and the important features. Out of what we'll end up with, we will define a clear objective to this project. (Note that this work was first done without any assumptions, only at the end did I choose to focus on specific features when I did understand what direction I wanted to explore here.)

1.2 Cleaning and Pre-processing

```
[ ]: kes = kes.drop(['kepoi_name', 'kepid'], axis=1, inplace=False)        #Names/id␣
      ↪we don't need, we'll keep kepler_name
     kes = kes.drop(['koi_datalink_dvr'], axis=1, inplace=False)                ␣
      ↪#Links to report, useless here
     kes = kes.drop([c for c in kes.columns if kes.loc[:,c].isna().all()], axis=1,␣
      ↪inplace=False)     #cleaning NA columns
```

```
[ ]: #Checking that these columns are not really adding any information
     print((kes['koi_vet_stat'] == 'Done').all())
     print((kes['koi_vet_date'] == '2018-08-16').all())
     print((kes['koi_disp_prov'] == 'q1_q17_dr25_sup_koi').all())
     print((kes['koi_parm_prov'] == 'q1_q17_dr25_sup_koi').all())
     print((kes['koi_tce_delivname'] == 'q1_q17_dr25_tce').all())
```

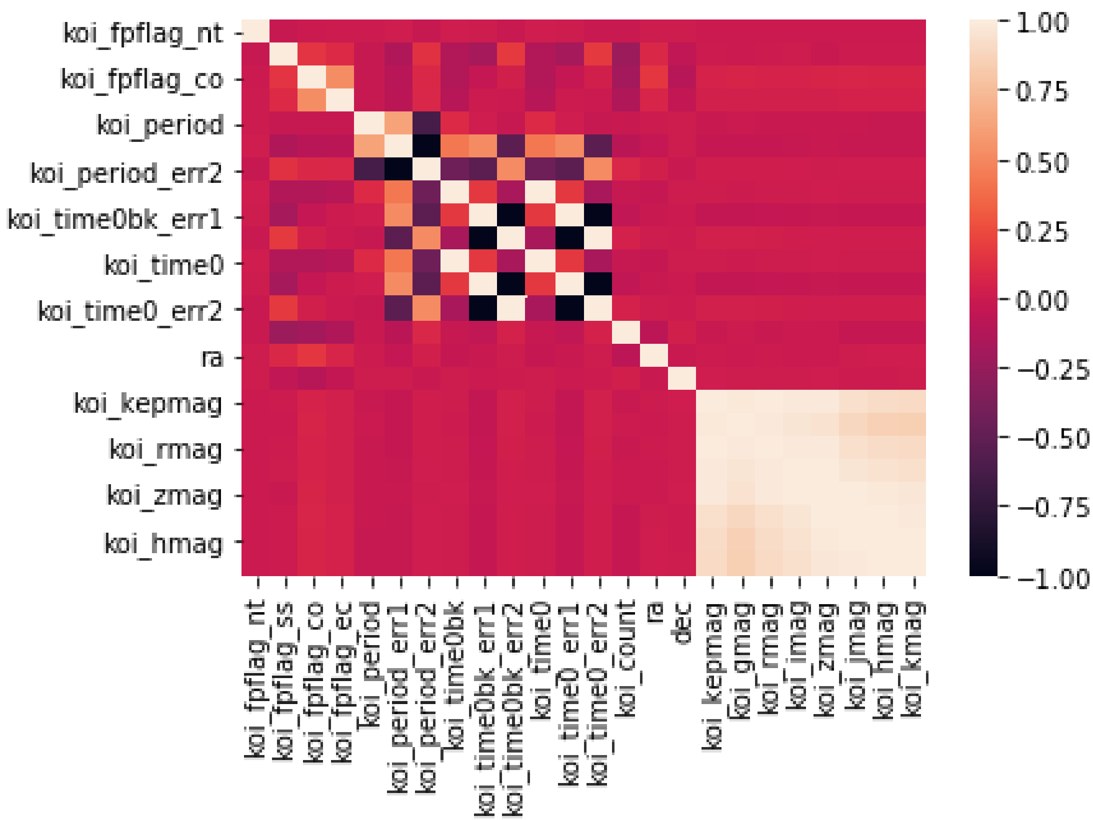
```
kes = kes.drop(['koi_vet_stat', 'koi_vet_date', 'koi_disp_prov',
 'koi_parm_prov', 'koi_tce_delivname'], axis=1, inplace=False)
```

```
True
True
True
True
True
```

Let's observe the correlation, if too strong we'll drop them (one of each pair), as only one provide new information and repetition might endanger our prediction and complexify the model and analysis.

[ ]:
```
sns.heatmap(kes.corr())
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1f5af910>
```



[ ]:
```
print((kes['koi_period_err1'].dropna() == -kes['koi_period_err2'].dropna()).
 all())
```

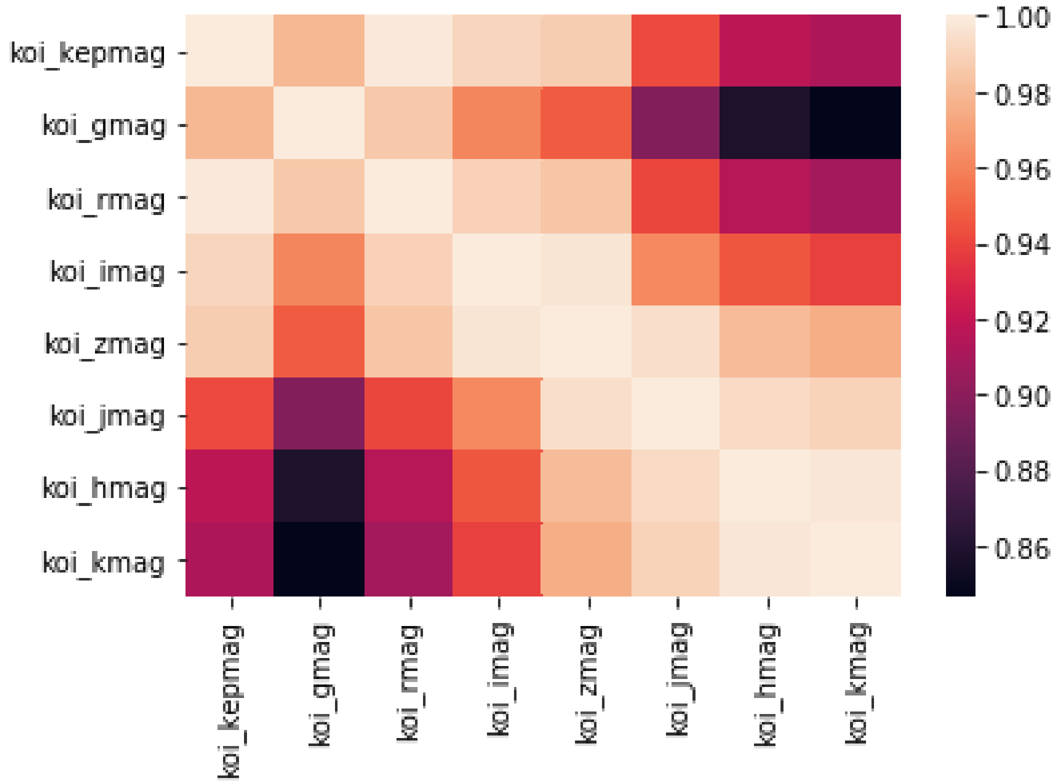
```
print((kes['koi_time0bk_err1'].dropna() == -kes['koi_time0bk_err2'].dropna()).
 ↪all())
print((kes['koi_time0_err1'].dropna() == -kes['koi_time0_err2'].dropna()).all())
print((kes['koi_time0bk_err1'].dropna() == kes['koi_time0_err1'].dropna()).
 ↪all())

kes = kes.drop(['koi_period_err2', 'koi_time0_err2', 'koi_time0bk',␣
 ↪'koi_time0bk_err1', 'koi_time0bk_err2'], axis=1, inplace=False)
```

```
True
True
True
True
```

The bright-colored square in the left bottom is suspect. Let's widen it.

[ ]:
```
sns.heatmap(kes.loc[:,['koi_kepmag', 'koi_gmag', 'koi_rmag', 'koi_imag',␣
 ↪'koi_zmag', 'koi_jmag', 'koi_hmag', 'koi_kmag']].corr())
```

[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1f6adf10>



[ ]:
```
kes = kes.drop(['koi_gmag', 'koi_rmag', 'koi_imag', 'koi_zmag', 'koi_jmag',␣
↪'koi_hmag', 'koi_kmag'], axis=1, inplace=False)
```

[ ]:
```
for c in ['koi_fpflag_nt', 'koi_fpflag_ss', 'koi_fpflag_co', 'koi_fpflag_ec',␣
↪'koi_count']:
    print(c, kes[c].unique())
```

```
koi_fpflag_nt [  0   1 465]
koi_fpflag_ss [0 1]
koi_fpflag_co [0 1]
koi_fpflag_ec [0 1]
koi_count [1 2 3 5 6 4 7]
```

There's only one row where 'koi_fpflag_nt'==465, so let's drop it, it must be an error on the data.

[ ]:
```
print(kes[kes['koi_fpflag_nt']==465].shape[0])
kes = kes.drop(645, axis=0, inplace=False)
```

```
1
```

Also, we saw that 'koi_count' has seven different values, because there should not be a clear order in the space of prediction for this feature (there is nothing that ensures that having 7 planets candidate in the system (this is the 'koi_count' feature) is 7 times more impacting than having only one). To deal with that, we one-hot encode this feature.

[ ]:
```
OHE = OneHotEncoder()

X = OHE.fit_transform(kes[['koi_count']])
kes[['koi_count_' + i.astype(str) for i in OHE.categories_[0]]] = pd.
↪DataFrame(X.toarray())
kes = kes.drop(['koi_count'], axis=1, inplace=False)
kes.head()
```

[ ]:
```
      kepler_name koi_disposition koi_pdisposition  koi_fpflag_nt  \
0     Kepler-1 b       CONFIRMED        CANDIDATE              0
1     Kepler-2 b       CONFIRMED        CANDIDATE              0
2     Kepler-3 b       CONFIRMED        CANDIDATE              0
3  Kepler-1658 b       CONFIRMED        CANDIDATE              0
4            NaN       CANDIDATE        CANDIDATE              0

   koi_fpflag_ss  koi_fpflag_co  koi_fpflag_ec  koi_period  koi_period_err1  \
0              0              0              0    2.470613     2.700000e-08
1              1              0              0    2.204735     4.300000e-08
2              0              0              0    4.887803     4.660000e-07
3              0              0              0    3.849372     2.344000e-06
4              0              0              0    4.780328     8.520000e-07

     koi_time0  koi_time0_err1         ra        dec  koi_kepmag  koi_count_1  \
```

```
0  2454955.763        0.000009  286.80847  49.316399      11.338           1.0
1  2454954.359        0.000016  292.24728  47.969521      10.463           1.0
2  2454957.813        0.000075  297.70935  48.080853       9.174           1.0
3  2454990.527        0.000488  294.35654  38.947380      11.432           1.0
4  2454965.974        0.000148  289.73972  44.647419      11.665           0.0

   koi_count_2  koi_count_3  koi_count_4  koi_count_5  koi_count_6  \
0          0.0          0.0          0.0          0.0          0.0
1          0.0          0.0          0.0          0.0          0.0
2          0.0          0.0          0.0          0.0          0.0
3          0.0          0.0          0.0          0.0          0.0
4          1.0          0.0          0.0          0.0          0.0

   koi_count_7
0          0.0
1          0.0
2          0.0
3          0.0
4          0.0
```

[ ]:
```
kes.loc[kes['kepler_name'].isnull(), 'kepler_name'] = 'no_name'
```

[ ]:
```
print(kes['koi_disposition'].unique())          #Actual
print(kes['koi_pdisposition'].unique())
```

```
['CONFIRMED' 'CANDIDATE' 'FALSE POSITIVE']
['CANDIDATE' 'FALSE POSITIVE']
```

Objective: Predict the 'koi_pdisposition' column, ie. the we'll try to predict if a planet could be habitanble ('CANDIDATE') or definitely not ('FALSE POSITIVE'). This column was created by physicists using Kepler's data. We'll do the same, but instead of using physical laws, we will try to implement a model that replicate this model (not as a formula but as predictions). We'll face a classification problem were this issue is to classify a stellar object as a 'CANDIDATE' or 'NON CANDIDATE' to be a habitable exoplanet.

Note: we cannot use the 'koi_disposition' column because it is basically the same as 'koi_pdisposition' were 'CANDIDATE's who were verified were changed to 'CONFIRMED'. So it does not really make sense to try and predict a 'CONFIRMED' as it can only be confirmed by more direct observations. Here we'll focus on eliminating candidates, as we usually want with physical laws. Ie, we'll reproduce the 'koi_pdisposition' column.

[ ]:
```
kes = kes.drop(['koi_disposition'], axis=1, inplace=False)
kes.head()
```

[ ]:
```
      kepler_name koi_pdisposition  koi_fpflag_nt  koi_fpflag_ss  \
0     Kepler-1 b        CANDIDATE              0              0
1     Kepler-2 b        CANDIDATE              0              1
2     Kepler-3 b        CANDIDATE              0              0
```

```
3  Kepler-1658 b        CANDIDATE              0              0
4        no_name        CANDIDATE              0              0

   koi_fpflag_co  koi_fpflag_ec  koi_period  koi_period_err1    koi_time0  \
0              0              0    2.470613     2.700000e-08  2454955.763
1              0              0    2.204735     4.300000e-08  2454954.359
2              0              0    4.887803     4.660000e-07  2454957.813
3              0              0    3.849372     2.344000e-06  2454990.527
4              0              0    4.780328     8.520000e-07  2454965.974

   koi_time0_err1         ra        dec  koi_kepmag  koi_count_1  koi_count_2  \
0        0.000009  286.80847  49.316399      11.338          1.0          0.0
1        0.000016  292.24728  47.969521      10.463          1.0          0.0
2        0.000075  297.70935  48.080853       9.174          1.0          0.0
3        0.000488  294.35654  38.947380      11.432          1.0          0.0
4        0.000148  289.73972  44.647419      11.665          0.0          1.0

   koi_count_3  koi_count_4  koi_count_5  koi_count_6  koi_count_7
0          0.0          0.0          0.0          0.0          0.0
1          0.0          0.0          0.0          0.0          0.0
2          0.0          0.0          0.0          0.0          0.0
3          0.0          0.0          0.0          0.0          0.0
4          0.0          0.0          0.0          0.0          0.0
```

One good news is, the classification problem seems to be well distributed as we have approwimately the same number of examples of each label.

```
[ ]: print( f"There is {kes[kes['koi_pdisposition'] == 'CANDIDATE'].shape[0]/kes.
     ↪shape[0]:.2f}% of 'CANDIDATE', and {kes[kes['koi_pdisposition'] == 'FALSE␣
     ↪POSITIVE'].shape[0]/kes.shape[0]:.2f}% of 'FALSE POSITIVE'")
```

```
There is 0.49% of 'CANDIDATE', and 0.51% of 'FALSE POSITIVE'
```

Finally, let's create the train and test sets.

```
[ ]: data_train, data_test = train_test_split(kes, test_size=0.25, shuffle=True,␣
     ↪random_state=42)
```

2 EDA

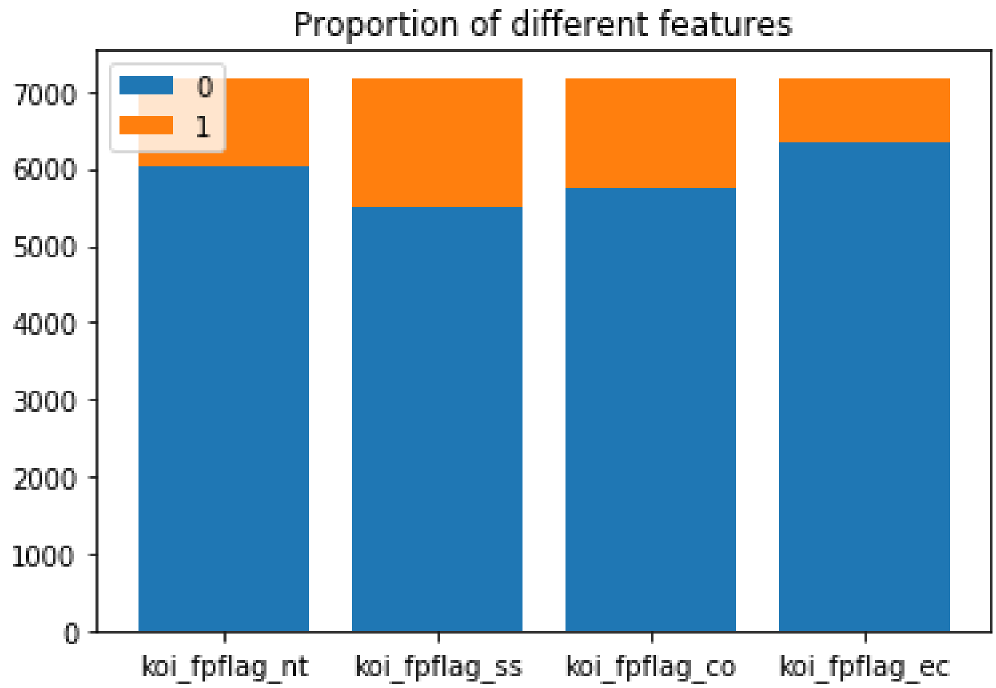
Now, time for digging in depth in the data structuration. We'll try to understand or at least extract some relationships by visualizing the different features under different angles.

```
[ ]: data_train.head()
```

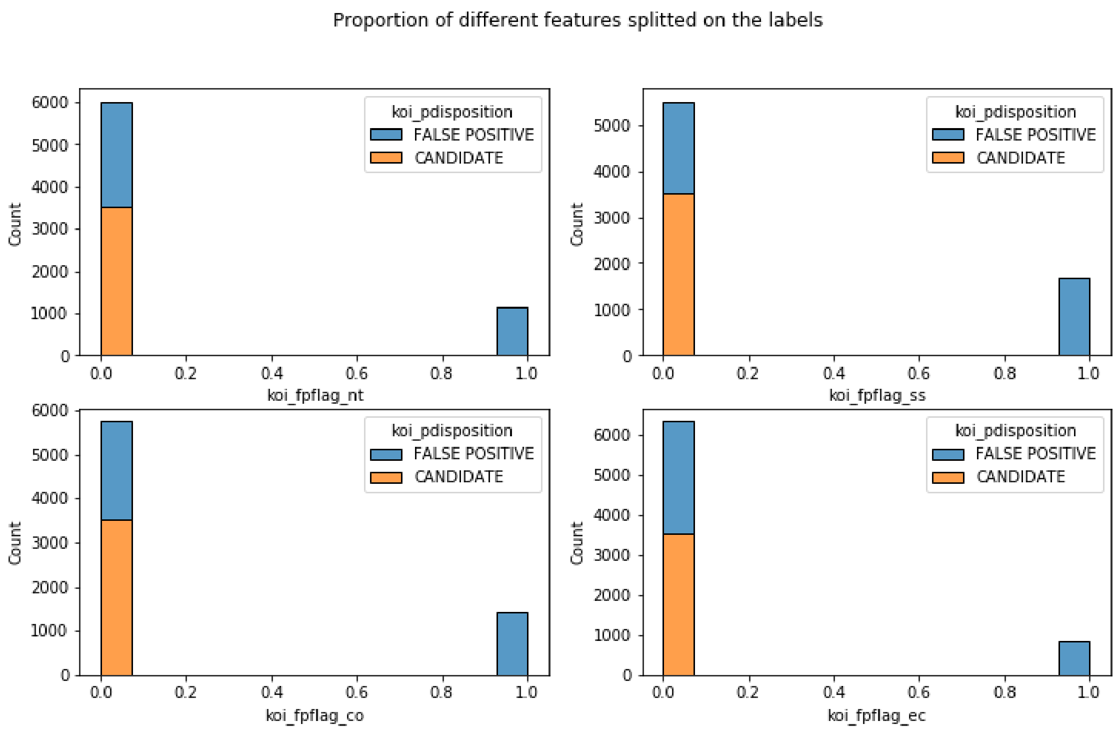
```
[ ]:     kepler_name koi_pdisposition  koi_fpflag_nt  koi_fpflag_ss  \
     1386     no_name   FALSE POSITIVE              0              1
     655      no_name   FALSE POSITIVE              0              1
```

```
1202     no_name        CANDIDATE              0              0
7466     no_name        CANDIDATE              0              0
7321     no_name   FALSE POSITIVE              0              1

      koi_fpflag_co  koi_fpflag_ec  koi_period  koi_period_err1    koi_time0  \
1386              0              0   12.713794     2.600000e-07  2455008.073
655               1              1   17.907661     3.764000e-05  2455008.560
1202              0              0    6.546261     2.668000e-05  2454966.860
7466              0              0  154.536881     5.439000e-03  2455010.697
7321              0              0    7.244779     3.890000e-07  2454964.860

      koi_time0_err1         ra        dec  koi_kepmag  koi_count_1  \
1386        0.000015  287.55972  46.957085      13.102          1.0
655         0.001760  296.10959  51.005501      14.162          1.0
1202        0.003170  292.23676  41.085880      15.855          0.0
7466        0.057000  299.87131  46.822491      13.654          1.0
7321        0.000044  294.48996  38.199619      13.926          0.0

      koi_count_2  koi_count_3  koi_count_4  koi_count_5  koi_count_6  \
1386          0.0          0.0          0.0          0.0          0.0
655           0.0          0.0          0.0          0.0          0.0
1202          0.0          0.0          0.0          1.0          0.0
7466          0.0          0.0          0.0          0.0          0.0
7321          1.0          0.0          0.0          0.0          0.0

      koi_count_7
1386          0.0
655           0.0
1202          0.0
7466          0.0
7321          0.0
```

[ ]:
```
binary_variables = ['koi_fpflag_nt', 'koi_fpflag_ss', 'koi_fpflag_co', 'koi_fpflag_ec']
plt.bar(binary_variables, [data_train.groupby(b).count().loc[0, 'kepler_name'] for b in binary_variables])
plt.bar(binary_variables, [data_train.groupby(b).count().loc[1, 'kepler_name'] for b in binary_variables], bottom=[data_train.groupby(b).count().loc[0, 'kepler_name'] for b in binary_variables])
plt.legend([0, 1])
plt.title('Proportion of different features')
plt.show()
```



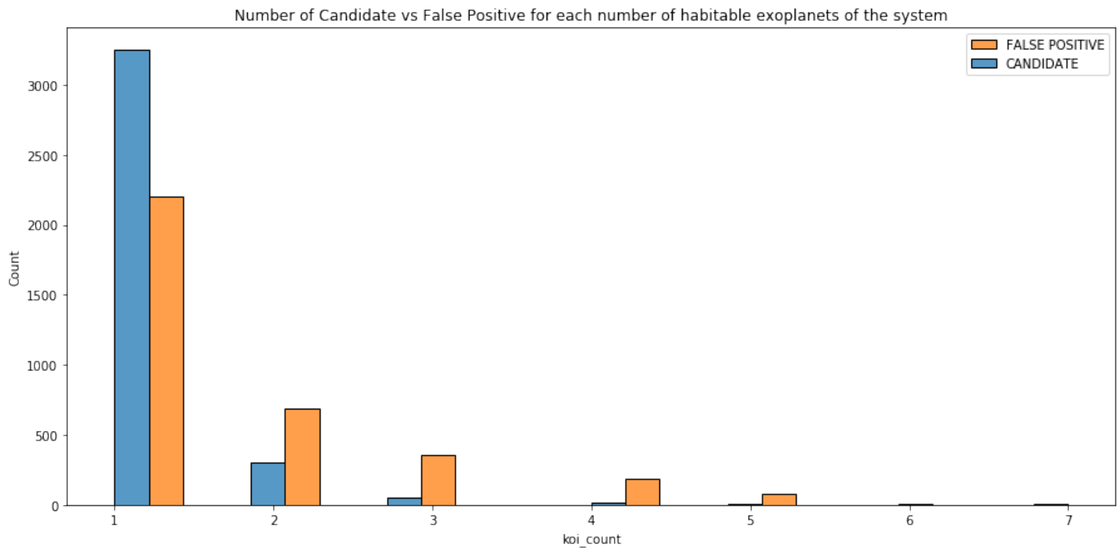
[ ]:
```
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=((12,7)))
sns.histplot(ax=ax[0, 0], data=data_train, x='koi_fpflag_nt',␣
 ↪hue='koi_pdisposition', multiple='stack')
sns.histplot(ax=ax[0, 1], data=data_train, x='koi_fpflag_ss',␣
 ↪hue='koi_pdisposition', multiple='stack')
sns.histplot(ax=ax[1, 0], data=data_train, x='koi_fpflag_co',␣
 ↪hue='koi_pdisposition', multiple='stack')
sns.histplot(ax=ax[1, 1], data=data_train, x='koi_fpflag_ec',␣
 ↪hue='koi_pdisposition', multiple='stack')
plt.suptitle('Proportion of different features splitted on the labels')
plt.show()
```



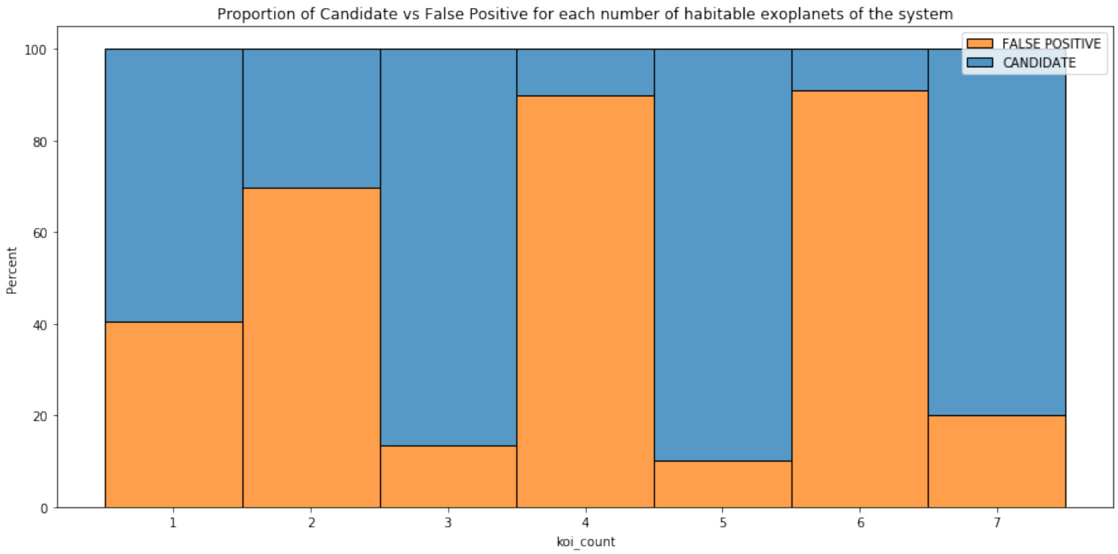

```
[ ]: data_train['koi_count'] = 0
     for i in range(1,8):
         data_train['koi_count'] += data_train['koi_count_'+str(i)]*i

     plt.figure(figsize=((15,7)))
     sns.histplot(data=data_train, x='koi_count', hue='koi_pdisposition',␣
      ↪multiple='dodge')
     plt.legend(data_train['koi_pdisposition'].unique())
     plt.title('Number of Candidate vs False Positive for each number of habitable␣
      ↪exoplanets of the system')
     plt.show()
```

```
/Users/adriensade/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/Users/adriensade/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```



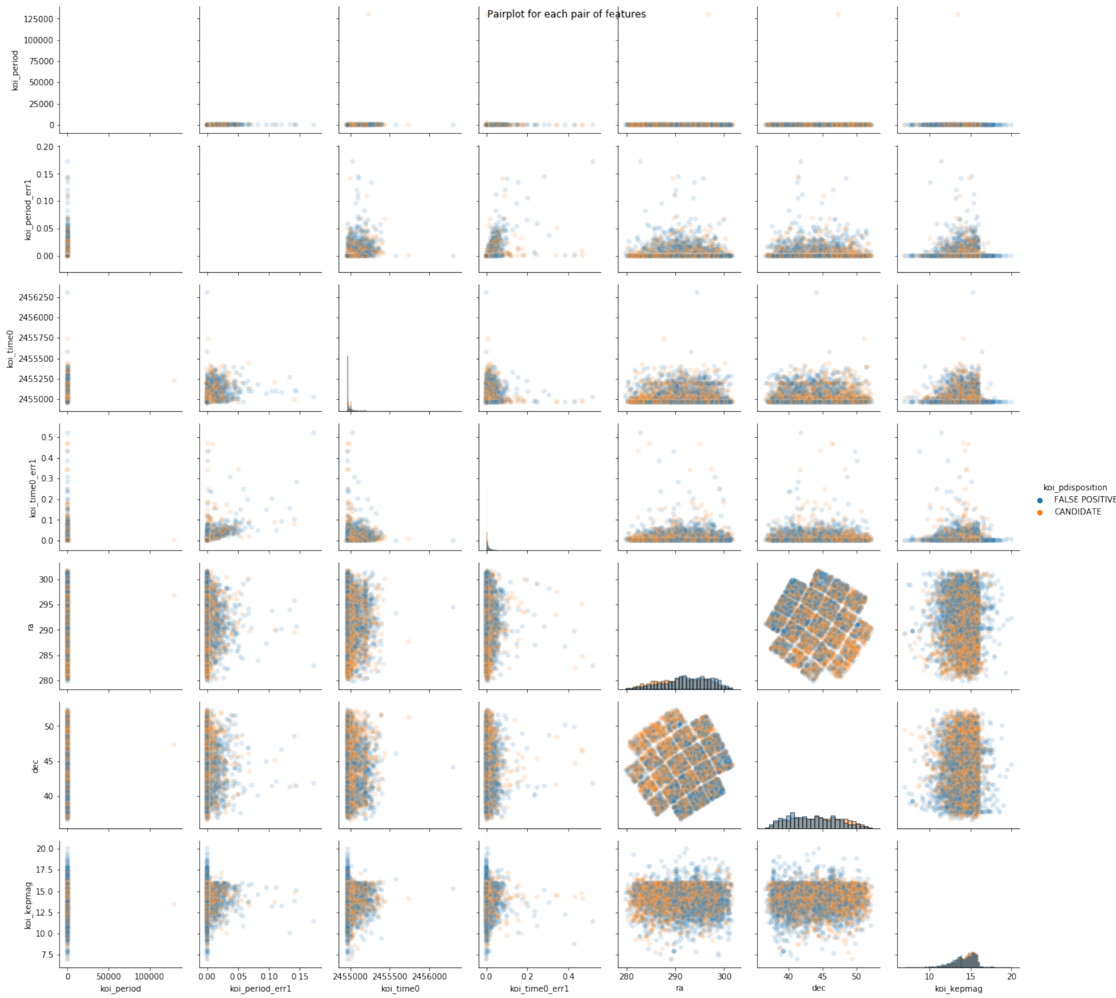
[ ]:
```
plt.figure(figsize=((15,7)))
for i in range(1,8):
    sns.histplot(data=data_train[data_train['koi_count']==i], x='koi_count',␣
 ↪hue='koi_pdisposition', multiple='stack', stat='percent')
plt.legend(data_train['koi_pdisposition'].unique())
plt.title('Proportion of Candidate vs False Positive for each number of␣
 ↪habitable exoplanets of the system')
plt.show()

data_train = data_train.drop('koi_count', axis=1, inplace=False)
```




```
[ ]: continuous_variables = ['koi_period', 'koi_period_err1', 'koi_time0',␣
     ↪'koi_time0_err1', 'ra', 'dec', 'koi_kepmag']
```

```
[ ]: g = sns.PairGrid(data=data_train, vars=continuous_variables,␣
     ↪hue='koi_pdisposition')
     g.map_diag(sns.histplot)
     g.map_offdiag(sns.scatterplot, alpha=0.15)
     g.add_legend()
     g.fig.suptitle('Pairplot for each pair of features')
```

```
[ ]: Text(0.5, 0.98, 'Pairplot for each pair of features')
```

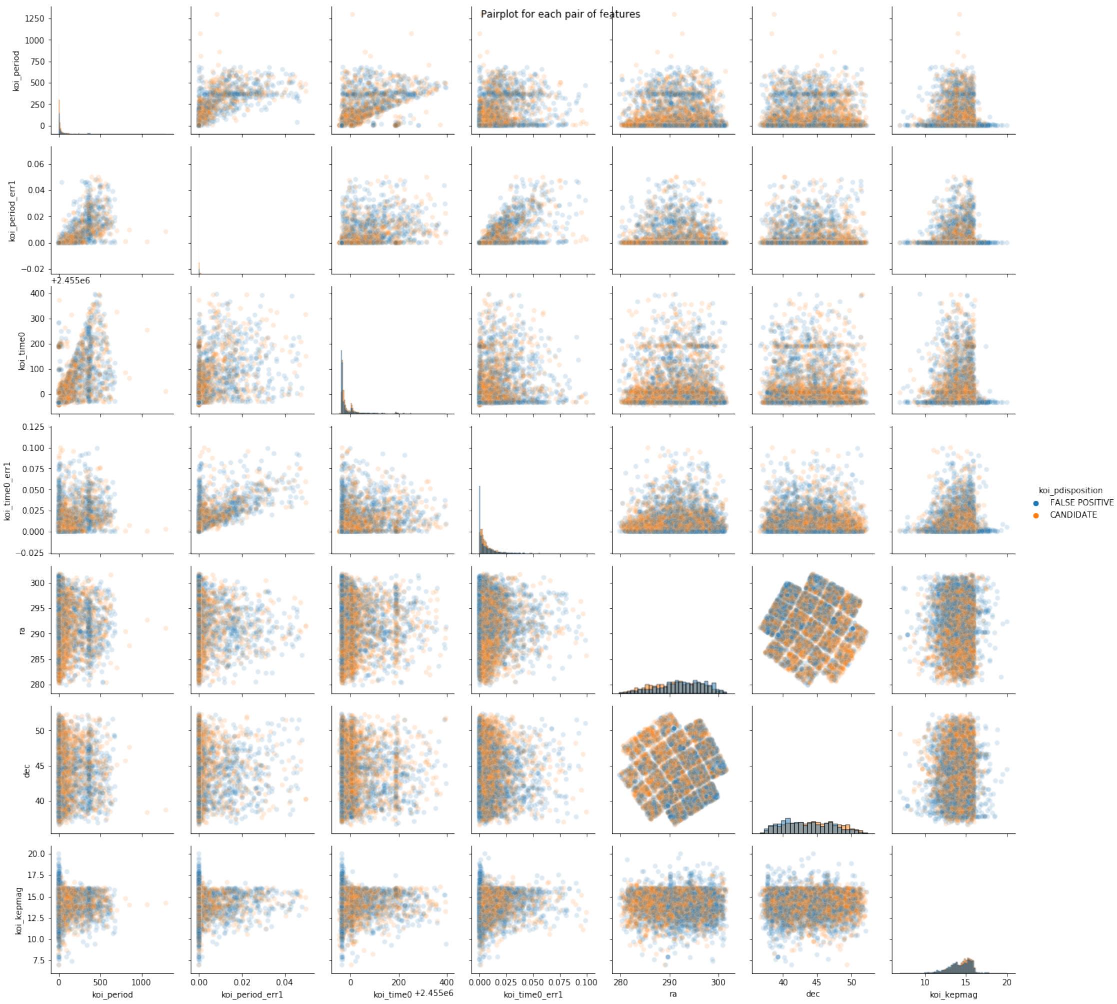


Because we have many examples, we can get rid of the outliers rather roughly.

[ ]:
```
data_train = data_train[data_train['koi_period'] < 5000]
data_train = data_train[data_train['koi_period_err1'] < 0.05]
data_train = data_train[data_train['koi_time0'] < 2.4554e6]
data_train = data_train[data_train['koi_time0_err1'] < 0.1]
```

[ ]:
```
g = sns.PairGrid(data=data_train, vars=continuous_variables,␣
  ↪hue='koi_pdisposition')
g.map_diag(sns.histplot)
g.map_offdiag(sns.scatterplot, alpha=0.15)
g.add_legend()
g.fig.suptitle('Pairplot for each pair of features')
```

[ ]:
```
Text(0.5, 0.98, 'Pairplot for each pair of features')
```



There seems to be a few exponential-relationships.

```
[ ]: data_train['koi_period'] = np.log(data_train['koi_period'])
     data_train['koi_period_err1'] = np.log(data_train['koi_period_err1'])
     data_train['koi_time0'] = np.log(data_train['koi_time0'])
     data_train['koi_time0_err1'] = np.log(data_train['koi_time0_err1'])
```

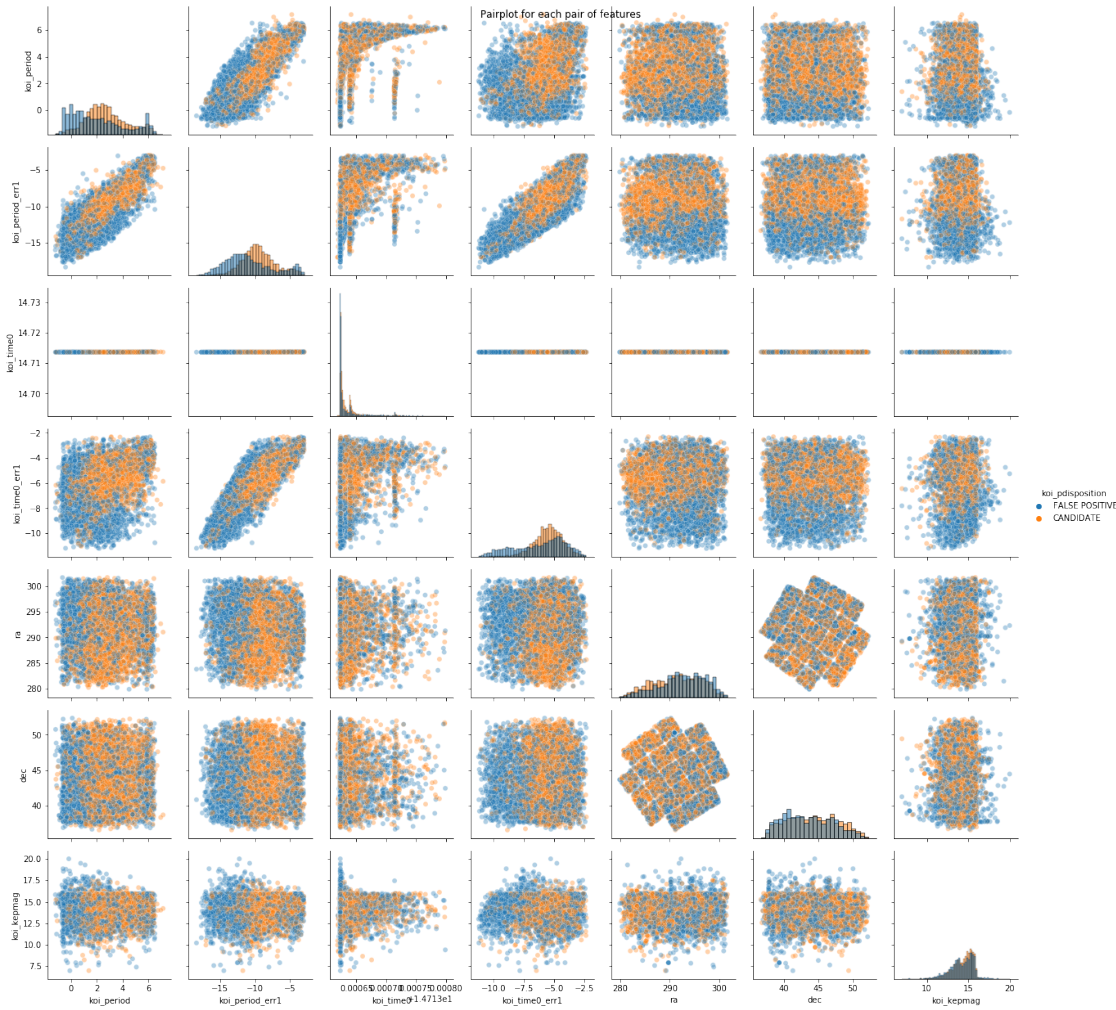
```
[ ]: data_test['koi_period'] = np.log(data_test['koi_period'])
     data_test['koi_period_err1'] = np.log(data_test['koi_period_err1'])
     data_test['koi_time0'] = np.log(data_test['koi_time0'])
     data_test['koi_time0_err1'] = np.log(data_test['koi_time0_err1'])
```

```
/Users/adriensade/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
/Users/adriensade/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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

/Users/adriensade/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
/Users/adriensade/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  after removing the cwd from sys.path.
```

[ ]:
```
g = sns.PairGrid(data=data_train, vars=continuous_variables,␣
 ↪hue='koi_pdisposition')
g.map_diag(sns.histplot)
g.map_offdiag(sns.scatterplot, alpha=0.35)
g.add_legend()
g.fig.suptitle('Pairplot for each pair of features')
```

[ ]:
```
Text(0.5, 0.98, 'Pairplot for each pair of features')
```



Finally, let's standardize the continuous variables before applying ML models.

```
[ ]: Standardizer = StandardScaler()
     data_train[continuous_variables] = Standardizer.fit_transform(data_train[continuous_variables])
     data_test[continuous_variables] = Standardizer.transform(data_test[continuous_variables])
```

```
/Users/adriensade/opt/anaconda3/lib/python3.7/site-
packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
```

```
/Users/adriensade/opt/anaconda3/lib/python3.7/site-
packages/pandas/core/indexing.py:965: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self.obj[item] = s
```

[ ]:
```python
data_train = data_train.dropna()
data_test = data_test.dropna()
```

[ ]:
```python
data_train.head()
```

[ ]:
```
     kepler_name koi_pdisposition  koi_fpflag_nt  koi_fpflag_ss  \
1386     no_name   FALSE POSITIVE              0              1
655      no_name   FALSE POSITIVE              0              1
1202     no_name        CANDIDATE              0              0
7466     no_name        CANDIDATE              0              0
7321     no_name   FALSE POSITIVE              0              1

      koi_fpflag_co  koi_fpflag_ec  koi_period  koi_period_err1  koi_time0  \
1386              0              0    0.081010        -1.664926   0.169511
655               1              1    0.264962        -0.024238   0.177237
1202              0              0   -0.275461        -0.137732  -0.484321
7466              0              0    1.422353         1.615836   0.211140
7321              0              0   -0.221014        -1.532059  -0.516051

      koi_time0_err1        ra       dec  koi_kepmag  koi_count_1  \
1386       -2.997916 -0.942725  0.866710   -0.869528          1.0
655        -0.290962  0.849668  1.985977   -0.093046          1.0
1202        0.043312  0.037769 -0.756505    1.147129          0.0
7466        1.684702  1.638275  0.829498   -0.465171          1.0
7321       -2.382713  0.510130 -1.554471   -0.265923          0.0

      koi_count_2  koi_count_3  koi_count_4  koi_count_5  koi_count_6  \
1386          0.0          0.0          0.0          0.0          0.0
655           0.0          0.0          0.0          0.0          0.0
1202          0.0          0.0          0.0          1.0          0.0
7466          0.0          0.0          0.0          0.0          0.0
7321          1.0          0.0          0.0          0.0          0.0

      koi_count_7
1386          0.0
655           0.0
1202          0.0
7466          0.0
```

```
7321            0.0
```

2.1 Prediction Model

Now that we have the data fully prepared, we can apply some classification models. To have a first glimpse of which will work best, let's run a lazypredict on this and see which algorithms fit best. Then, we'll move forward using the bests of them.

[ ]:
```
X_train, Y_train = data_train.drop(['kepler_name', 'koi_pdisposition'], axis=1,␣
↪inplace=False), data_train['koi_pdisposition']
X_test, Y_test = data_test.drop(['kepler_name', 'koi_pdisposition'], axis=1,␣
↪inplace=False), data_test['koi_pdisposition']
```

[ ]:
```
#DO NOT RUN!!!
from lazypredict.Supervised import LazyClassifier

LPC = LazyClassifier()

models, predictions = LPC.fit(X_train.astype(float), X_test.astype(float),␣
↪Y_train, Y_test)

models.head(30)
```

```
100%|     | 29/29 [00:15<00:00,  1.85it/s]
```

[ ]:

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	\
LinearSVC	0.99	0.99	None	0.99	
ExtraTreesClassifier	0.99	0.99	None	0.99	
XGBClassifier	0.99	0.99	None	0.99	
RandomForestClassifier	0.99	0.99	None	0.99	
LogisticRegression	0.99	0.99	None	0.99	
GaussianNB	0.99	0.99	None	0.99	
LGBMClassifier	0.99	0.99	None	0.99	
CalibratedClassifierCV	0.99	0.99	None	0.99	
AdaBoostClassifier	0.99	0.99	None	0.99	
BaggingClassifier	0.99	0.99	None	0.99	
SVC	0.99	0.99	None	0.99	
Perceptron	0.99	0.99	None	0.99	
SGDClassifier	0.99	0.99	None	0.99	
LabelSpreading	0.99	0.99	None	0.99	
LabelPropagation	0.98	0.99	None	0.98	
DecisionTreeClassifier	0.98	0.98	None	0.98	
PassiveAggressiveClassifier	0.98	0.98	None	0.98	
KNeighborsClassifier	0.98	0.98	None	0.98	
NuSVC	0.98	0.98	None	0.98	
RidgeClassifier	0.97	0.97	None	0.97	

RidgeClassifierCV	0.97	0.97	None	0.97
LinearDiscriminantAnalysis	0.97	0.97	None	0.97
ExtraTreeClassifier	0.97	0.97	None	0.97
BernoulliNB	0.96	0.96	None	0.96
NearestCentroid	0.89	0.89	None	0.89
QuadraticDiscriminantAnalysis	0.82	0.82	None	0.82
DummyClassifier	0.49	0.50	None	0.33

	Time Taken
Model	
LinearSVC	0.36
ExtraTreesClassifier	0.37
XGBClassifier	0.49
RandomForestClassifier	0.91
LogisticRegression	0.11
GaussianNB	0.06
LGBMClassifier	0.29
CalibratedClassifierCV	1.30
AdaBoostClassifier	0.92
BaggingClassifier	0.37
SVC	0.29
Perceptron	0.07
SGDClassifier	0.09
LabelSpreading	2.98
LabelPropagation	2.21
DecisionTreeClassifier	0.10
PassiveAggressiveClassifier	0.07
KNeighborsClassifier	0.72
NuSVC	3.28
RidgeClassifier	0.08
RidgeClassifierCV	0.10
LinearDiscriminantAnalysis	0.11
ExtraTreeClassifier	0.05
BernoulliNB	0.07
NearestCentroid	0.07
QuadraticDiscriminantAnalysis	0.08
DummyClassifier	0.05

The accuracy reachable is extremely high. That's because we are trying to recreate a prediction made by scientists without knowing the physical model (that is my point of view of someone who doen't know anything about planetary science), and not actually making a prediction for future events. For all we know, scientists might have used ML to create this column as well.

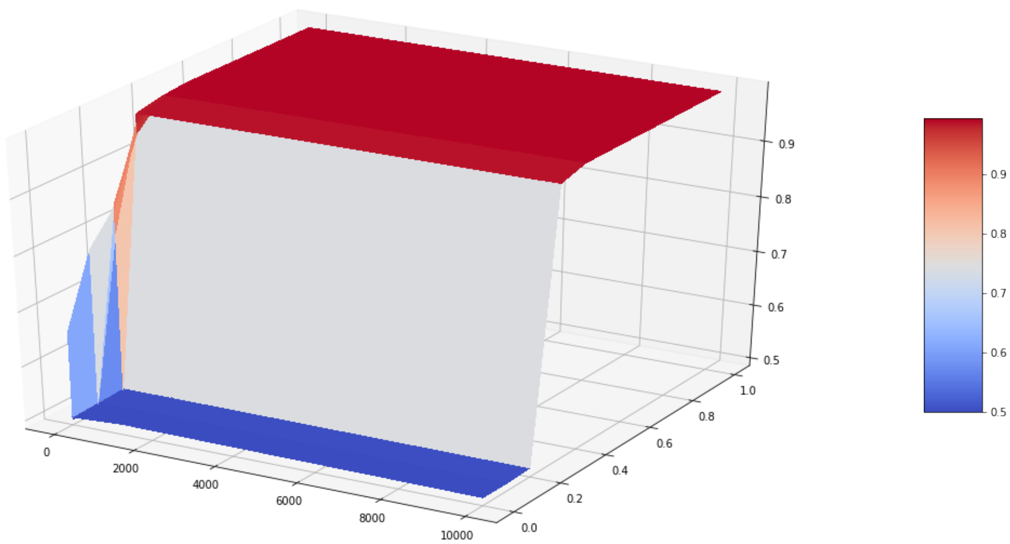
Many classifiers seem to work, I will focus on the simplest one seen in class as it seems to work pretty well: Logistic Regression. We'll use a cross validation (with a stratified K-fold strategy) to find the best parameters.

[ ]:
```
ratio_list = np.arange(0,1.02, step=0.1)
c_list = np.logspace(-5, 4, 11)

LR = LogisticRegression(penalty='elasticnet', solver='saga', max_iter=1000)

model = GridSearchCV(LR,
                    {'l1_ratio' : ratio_list, 'C' : c_list},
                    scoring = 'accuracy',
                    cv = 4,
                    refit = True)

model.fit(X_train, Y_train)
```

[ ]:
```
GridSearchCV(cv=4, error_score=nan,
             estimator=LogisticRegression(C=1.0, class_weight=None, dual=False,
                                          fit_intercept=True,
                                          intercept_scaling=1, l1_ratio=None,
                                          max_iter=1000, multi_class='auto',
                                          n_jobs=None, penalty='elasticnet',
                                          random_state=None, solver='saga',
                                          tol=0.0001, verbose=0,
                                          warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'C': array([1.00000000e-05, 7.94328235e-05,
6.30957344e-04, 5.01187234e-03,
       3.98107171e-02, 3.16227766e-01, 2.51188643e+00, 1.99526231e+01,
       1.58489319e+02, 1.25892541e+03, 1.00000000e+04]),
                         'l1_ratio': array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6,
0.7, 0.8, 0.9, 1. ])},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring='accuracy', verbose=0)
```

[ ]:
```
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

fig, ax = plt.subplots(subplot_kw={"projection": "3d"}, figsize=(20,10))

# Make data.
X = c_list
Y = ratio_list
X, Y = np.meshgrid(X, Y)
Z = model.cv_results_['mean_test_score'].reshape((len(X),len(Y)))
```

```
# Plot the surface.
surf = ax.plot_surface(X, Y, Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)

# Add a color bar which maps values to colors.
fig.colorbar(surf, shrink=0.5, aspect=5)
plt.title('')
plt.show()
```



Let's see how good is our final estimator.

[ ]:
```
LR_model = model.best_estimator_

Y_pred = LR_model.predict(X_test)

print(f'We reach {(Y_pred == Y_test).mean():.4f}% of accuracy on the test set')
```

```
We reach 0.9938% of accuracy on the test set
```

Let's dig a little bit, and look at the confusion matrix.

[ ]:
```
confusion_matrix = pd.DataFrame(confusion_matrix(Y_test, Y_pred),␣
 ↪columns=['Pred_CANDIDATE', 'Pred_FALSE_POSITIVE'], index=['CANDIDATE',␣
 ↪'FALSE_POSITIVE'])
confusion_matrix
```

[ ]:

	Pred_CANDIDATE	Pred_FALSE_POSITIVE
CANDIDATE	1136	3

FALSE_POSITIVE	11	1102