Final_Project_CS156

December 18, 2020

0.0.1 Import Necessary Packages and download the csv files

Importing packages

```
In [2]: # import packages
        import pandas as pd
        import pandas_profiling as pd_prof
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline

        # for downloading file
        from google.colab import files

        ## Use random forest to create and evaluate new model
        from sklearn.ensemble import RandomForestClassifier
```

```
In [3]: !pip install astroquery
```

```
Requirement already satisfied: astroquery in /usr/local/lib/python3.6/dist-packages (0.4.1)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from astroquery)
Requirement already satisfied: astropy>=3.1 in /usr/local/lib/python3.6/dist-packages (from as
Requirement already satisfied: requests>=2.4.3 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: html5lib>=0.999 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: beautifulsoup4>=4.3.2 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from astroquery
Requirement already satisfied: keyring>=4.0 in /usr/local/lib/python3.6/dist-packages (from as
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/pytho
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from re
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: webencodings in /usr/local/lib/python3.6/dist-packages (from ht
Requirement already satisfied: SecretStorage>=3.2; sys_platform == "linux" in /usr/local/lib/p
Requirement already satisfied: importlib-metadata>=1; python_version < "3.8" in /usr/local/lib
Requirement already satisfied: jeepney>=0.4.2; sys_platform == "linux" in /usr/local/lib/pytho
Requirement already satisfied: cryptography>=2.0 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/dist-packages (from impor
```

```
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.6/dist-packages (from cryp
Requirement already satisfied: pycparser in /usr/local/lib/python3.6/dist-packages (from cffi>=
```

```
In [4]: # import astroquery
        import astropy.units as u
        import astropy.coordinates as coord
        from astroquery.gaia import Gaia
        from astroquery.vizier import Vizier
```

```
Created TAP+ (v1.2.1) - Connection:
        Host: gea.esac.esa.int
        Use HTTPS: True
        Port: 443
        SSL Port: 443
Created TAP+ (v1.2.1) - Connection:
        Host: geadata.esac.esa.int
        Use HTTPS: True
        Port: 443
        SSL Port: 443
```

Finding GAIA Data

```
In [5]: ## making a GAIA cone_search of 30m radius around NGC3766 center

        coordinate = coord.SkyCoord.from_name('NGC3766')
        print(coordinate)
        radius = u.Quantity(0.8, u.deg)
        Gaia.ROW_LIMIT = -1
        j = Gaia.cone_search_async(coordinate, radius)
        r = j.get_results()
        print(type(r))
```

```
<SkyCoord (ICRS): (ra, dec) in deg
    (174.075, -61.615)>
INFO: Query finished. [astroquery.utils.tap.core]
<class 'astropy.table.table.Table'>
```

```
In [6]: ## save the ASCII table as a panadas dataframe
        all_stars = r.to_pandas()
        all_stars
```

```
Out[6]:                 solution_id  ...      dist
        0       1635721458409799680  ...  0.000741
        1       1635721458409799680  ...  0.001082
        2       1635721458409799680  ...  0.001105
        3       1635721458409799680  ...  0.001147
```

```
4       1635721458409799680  ...  0.001680
...                     ...  ...       ...
641173  1635721458409799680  ...  0.799999
641174  1635721458409799680  ...  0.799999
641175  1635721458409799680  ...  0.799999
641176  1635721458409799680  ...  0.799999
641177  1635721458409799680  ...  0.799999

[641178 rows x 97 columns]
```

Applying Filter:

```
In [160]: all_stars_filtered = all_stars[all_stars['parallax_over_error'] > 3]
```

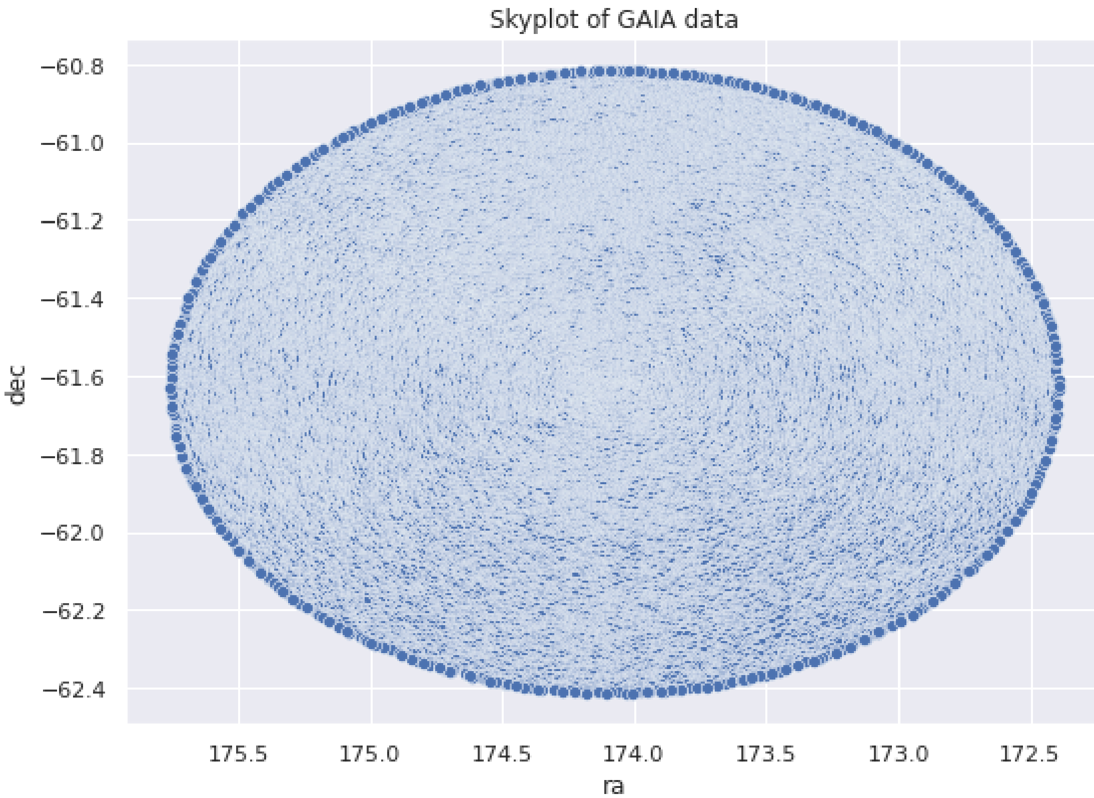
```
In [161]: all_stars_filtered = all_stars_filtered[(all_stars_filtered['pmdec_over_error'] > 3)
```

```
In [162]: all_stars_filtered.shape
```

```
Out[162]: (88492, 99)
```

Visualizing GAIA data

```
In [377]: ## plotting the skyplot

          sns.set(rc={'figure.figsize':(8.7,6.27)})

          skyplot = sns.scatterplot(x='ra', y='dec',
                                    data = all_stars_filtered)
          skyplot.invert_xaxis()
          plt.title('Skyplot of GAIA data')
          plt.show()
```



Finding Cantat Data

In [35]:
```
#### Finding Cantat catalogue

catalog_list = Vizier.find_catalogs('Cantat')
{k:v.description for k,v in catalog_list.items()}
```

Out[35]:
```
{'I/349': 'StarHorse, Gaia DR2 photo-astrometric distances (Anders+, 2019)',
 'J/A+A/561/A94': 'Velocities and photometry in Trumpler 20 (Donati+, 2014)',
 'J/A+A/564/A133': 'Gaia FGK benchmark stars: metallicity (Jofre+, 2014)',
 'J/A+A/569/A17': 'Gaia-ESO Survey: NGC6705 (Cantat-Gaudin+, 2014)',
 'J/A+A/582/A81': 'Gaia FGK benchmark stars: abundances (Jofre+, 2015)',
 'J/A+A/588/A120': 'Equivalent widths in 10 open clusters (Cantat-Gaudin+, 2016)',
 'J/A+A/591/A37': 'Gaia-ESO Survey. Parameters for cluster members (Jacobson+, 2016)'
 'J/A+A/597/A10': 'South Ecliptic Pole stars radial velocities (Fremat+, 2017)',
 'J/A+A/598/A68': 'Gaia-ESO Survey. Trumpler 23 (Overbeek+, 2017)',
 'J/A+A/601/A19': 'Gaia DR1 open cluster members (Gaia Collaboration+, 2017)',
 'J/A+A/603/A2': 'Gaia-ESO Survey abundances radial distribution (Magrini+, 2017)',
 'J/A+A/605/A79': 'TGAS Cepheids and RR Lyrae stars (Gaia Collaboration+, 2017)',
 'J/A+A/615/A49': 'TGAS stars membership in 128 open clusters (Cantat-Gaudin+, 2018)'
 'J/A+A/616/A10': '46 open clusters GaiaDR2 HR diagrams (Gaia Collaboration, 2018)',
 'J/A+A/616/A12': 'Gaia DR2 sources in GC and dSph (Gaia Collaboration+, 2018)',
```

```
'J/A+A/618/A59': 'Gaia DR2 confirmed new nearby open clusters (Castro-Ginard+, 2018)
'J/A+A/618/A93': 'Gaia DR2 open clusters in the Milky Way (Cantat-Gaudin+, 2018)',
'J/A+A/619/A155': 'Open cluster kinematics with Gaia DR2 (Soubiran+, 2018)',
'J/A+A/621/A115': 'Vela OB2 members (Cantat-Gaudin+, 2019)',
'J/A+A/623/A108': 'Age of 269 GDR2 open clusters (Bossini+, 2019)',
'J/A+A/623/A110': 'Gaia DR2. Variable stars in CMD (Gaia Collaboration+, 2019)',
'J/A+A/623/A80': 'Open clusters in APOGEE and GALAH surveys (Carrera+, 2019)',
'J/A+A/624/A126': 'New open clusters in Perseus direction (Cantat-Gaudin+, 2019)',
'J/A+A/626/A17': 'Young population in Vela-Puppis region (Cantat-Gaudin+, 2019)',
'J/A+A/627/A119': 'Extended halo of NGC 2682 (M 67) (Carrera+ 2019)',
'J/A+A/627/A35': 'New open clusters in Galactic anti-centre (Castro-Ginard+, 2019)',
'J/A+A/633/A99': 'Gaia DR2 open clusters in the Milky Way. II (Cantat-Gaudin+, 2020)
'J/A+A/635/A45': '570 new open clusters in the Galactic disc (Castro-Ginard+, 2020)'
'J/A+A/640/A1': 'Portrait Galactic disc (Cantat-Gaudin+, 2020)',
'J/MNRAS/446/1411': 'Trumpler 5 photometric BV catalog (Donati+, 2015)'}
```

In [36]:
```
## cheking the tables in the GAIA DR2 paper

Vizier.ROW_LIMIT = -1
#catalogs = Vizier.get_catalogs(catalog_list['J/A+A/633/A99'])
#catalogs
```

In [163]:
```
## saving only NGC 3766 data from Cantat GAIA DR2 paper

cantat_3766 = Vizier(catalog = 'J/A+A/633/A99/members', row_limit = -1).query_constra
cantat_3766 = cantat_3766[0].to_pandas()
cantat_3766
```

Out[163]:
```
        RA_ICRS    DE_ICRS  ...    _RA.icrs   _DE.icrs
0     174.195211 -61.479252  ...  174.195274 -61.479256
1     174.521895 -61.604049  ...  174.521956 -61.604052
2     174.447990 -61.635711  ...  174.448046 -61.635713
3     174.516866 -61.556410  ...  174.516927 -61.556414
4     174.439503 -61.377736  ...  174.439560 -61.377738
...          ...        ...  ...         ...        ...
1390  173.832089 -61.545743  ...  173.832147 -61.545747
1391  173.740258 -61.484316  ...  173.740318 -61.484319
1392  173.628115 -61.453283  ...  173.628173 -61.453290
1393  173.808609 -61.487716  ...  173.808671 -61.487721
1394  173.536850 -61.654957  ...  173.536916 -61.654964

[1395 rows x 13 columns]
```

In [164]:
```
# renaming the cantat table to match it with gaia_data
cantat_3766 = cantat_3766.rename(columns={'Source':'source_id',
                                          'Proba':'PMemb'})
```

In [165]:
```
# taking the subset of only source_id and PMemb
cantat_3766 = cantat_3766.loc[:,['source_id', 'PMemb']]
```

```
In [167]: # join the two table on source_id
          cantat_3766 = all_stars_filtered.join(cantat_3766.set_index('source_id'), on='source_
```

```
In [172]: # dropping the rows, where we don't have PMemb
          # (i.e. the source id was not in the cantat table)
          cantat_3766 = cantat_3766.dropna(subset=['PMemb'])
          cantat_3766
```

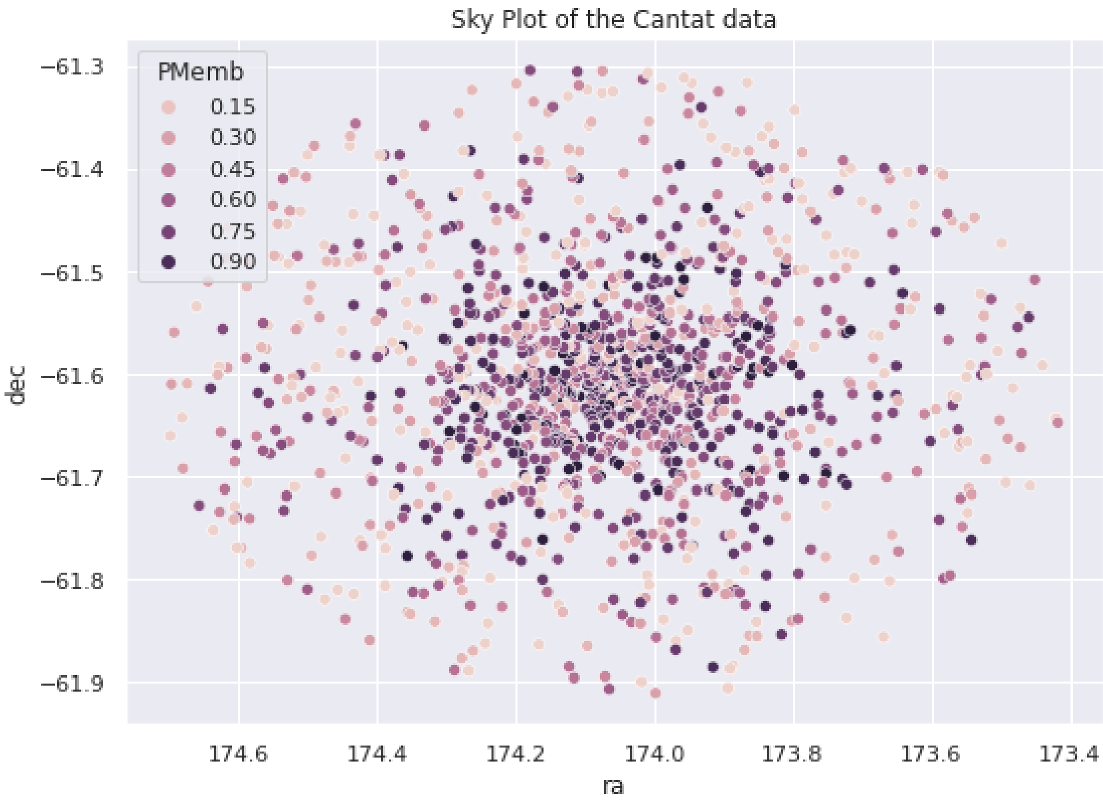
```
Out[172]:                solution_id  ... PMemb
          7       1635721458409799680  ...   1.0
          10      1635721458409799680  ...   1.0
          16      1635721458409799680  ...   0.6
          22      1635721458409799680  ...   0.5
          26      1635721458409799680  ...   0.4
          ...                     ...  ...   ...
          104667  1635721458409799680  ...   0.2
          104808  1635721458409799680  ...   0.7
          104811  1635721458409799680  ...   0.4
          104889  1635721458409799680  ...   0.1
          105689  1635721458409799680  ...   0.5

          [1345 rows x 100 columns]
```

```
In [49]: # saving both cantat and Gaia files as csv
         # if you want to save, comment out the next two lines

         cantat_3766.to_csv('NGC_3766_cantat.csv')
         # all_stars.to_csv('NGC_3766_Gaia_30m.csv')
```

```
In [50]: cantat_3766.describe()
```

```
Out[50]:         solution_id     source_id  ...  pmdec_over_error        PMemb
         count  1.358000e+03  1.358000e+03  ...       1358.000000  1358.000000
         mean   1.635721e+18  5.334617e+18  ...         16.334399     0.509423
         std    0.000000e+00  6.714607e+14  ...          8.465254     0.296726
         min    1.635721e+18  5.334149e+18  ...          1.530515     0.100000
         25%    1.635721e+18  5.334201e+18  ...          9.621796     0.200000
         50%    1.635721e+18  5.334209e+18  ...         14.757517     0.500000
         75%    1.635721e+18  5.335661e+18  ...         21.780832     0.800000
         max    1.635721e+18  5.335720e+18  ...         50.842703     1.000000

         [8 rows x 94 columns]
```

```
In [379]: ## plotting the skyplot

          skyplot = sns.scatterplot(x = cantat_3766['ra'], y = cantat_3766['dec'],
                                    hue = cantat_3766['PMemb'])
          skyplot.invert_xaxis()
          plt.title('Sky Plot of the Cantat data')
          plt.show()
```



0.0.2 Creating, Examining and Processing the Training Data

Training Data

```
In [199]: # import member dataset
          member = cantat_3766.copy()
```

```
In [381]: ### adding their distance from the center of the clusters

          ## NGC 3766
          center = coord.SkyCoord.from_name('NGC3766')
          center_ra, center_dec = center.ra.degree, center.dec.degree
          distance = np.sqrt( ((member['ra'] - center_ra)*np.cos(np.radians(member['dec'])))**2
          member['dist_3766_center'] = distance
```

```
In [383]: # maximum distance of stars in Cantat Data
          max(member.dist_3766_center)
```

```
Out[383]: 0.31621627751902387
```

```
In [380]: member['member'] = np.full(len(member), 1)
          member.head()
```

```
Out[380]:             solution_id                       designation  ...  dist_3766_center  member
          7   1635721458409799680  Gaia DR2 5334208304878429184  ...          0.003027       1
          10  1635721458409799680  Gaia DR2 5334208201799207680  ...          0.003206       1
          16  1635721458409799680  Gaia DR2 5334208197475777152  ...          0.004120       1
          22  1635721458409799680  Gaia DR2 5334208201799203072  ...          0.004735       1
          26  1635721458409799680  Gaia DR2 5334208407957797120  ...          0.005049       1

          [5 rows x 102 columns]
```

```
In [384]: ### adding their distance from the center of the clusters

          ## NGC 3766
          center = coord.SkyCoord.from_name('NGC3766')
          center_ra, center_dec = center.ra.degree, center.dec.degree
          distance = np.sqrt( ((all_stars_filtered['ra'] - center_ra)*np.cos(np.radians(all_sta
          all_stars_filtered['dist_3766_center'] = distance
```

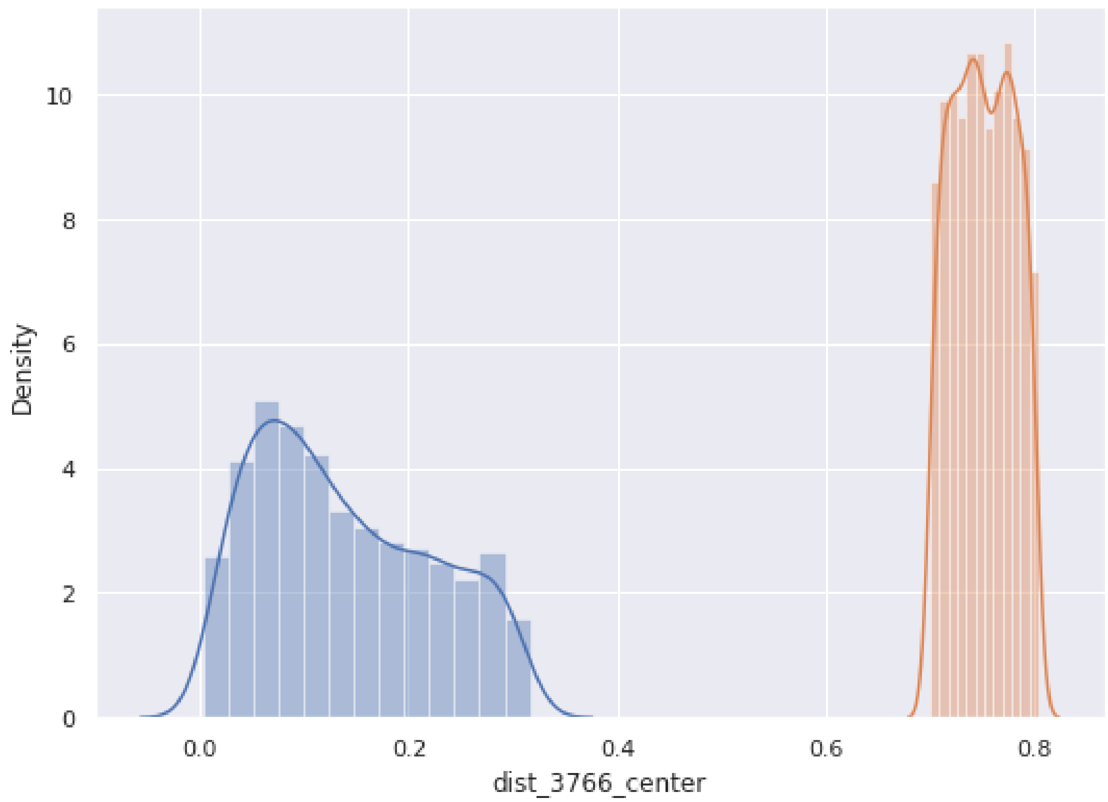
```
In [385]: non_member = all_stars_filtered[all_stars_filtered['dist_3766_center'] >= 0.7].sampl
```

```
In [386]: non_member['member'] = np.full(len(non_member), 0)
          non_member.head()
```

```
Out[386]:                 solution_id  ... member
          617823  1635721458409799680  ...      0
          530949  1635721458409799680  ...      0
          624419  1635721458409799680  ...      0
          547882  1635721458409799680  ...      0
          594412  1635721458409799680  ...      0

          [5 rows x 101 columns]
```

```
In [388]: sns.distplot(member['dist_3766_center'])
          sns.distplot(non_member['dist_3766_center'])
          plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
```



In [389]:
```
training_data = pd.concat([member, non_member])
```

In [390]:
```
# Examining the descriptive statistics of each column
training_data.describe()
```

Out[390]:

	solution_id	source_id	...	dist_3766_center	member
count	2.690000e+03	2.690000e+03	...	2690.000000	2690.000000
mean	1.635721e+18	5.334780e+18	...	0.445167	0.500000
std	0.000000e+00	7.120528e+14	...	0.312285	0.500093
min	1.635721e+18	5.333378e+18	...	0.003027	0.000000
25%	1.635721e+18	5.334205e+18	...	0.123652	0.000000
50%	1.635721e+18	5.334381e+18	...	0.508114	0.500000
75%	1.635721e+18	5.335672e+18	...	0.750425	1.000000
max	1.635721e+18	5.335930e+18	...	0.803653	1.000000

[8 rows x 96 columns]

In [392]:
```
# Choosing the features

feature_columns = ['parallax',
                   'pmra', 'pmdec']
```

```
features = training_data.loc[:,feature_columns]
targets = training_data['member']
```

In [393]:
```
# Dropping the NULL values from the using training set
# adding features and targets in a training set
training_set = pd.concat((features, targets), axis=1)

# dropping NA
training_set = training_set.dropna()

# finding where dtype is float64
float64_data = np.where(training_set.dtypes == 'float64')[0]

# change the data type to float32 from float64
training_set.iloc[:, float64_data] = training_set.iloc[:, float64_data].astype('floa


# set features, targets again
features, targets = training_set.iloc[:,:-1], training_set.iloc[:,-1]

features.describe()
```

Out[393]:

	parallax	pmra	pmdec
count	2690.000000	2690.000000	2690.000000
mean	0.558447	-7.096939	1.300540
std	0.499491	4.232746	2.496381
min	0.096016	-113.169159	-69.735741
25%	0.379664	-7.120129	0.846504
50%	0.456162	-6.734682	1.068744
75%	0.535136	-6.378384	1.648697
max	9.502661	19.480213	31.991152

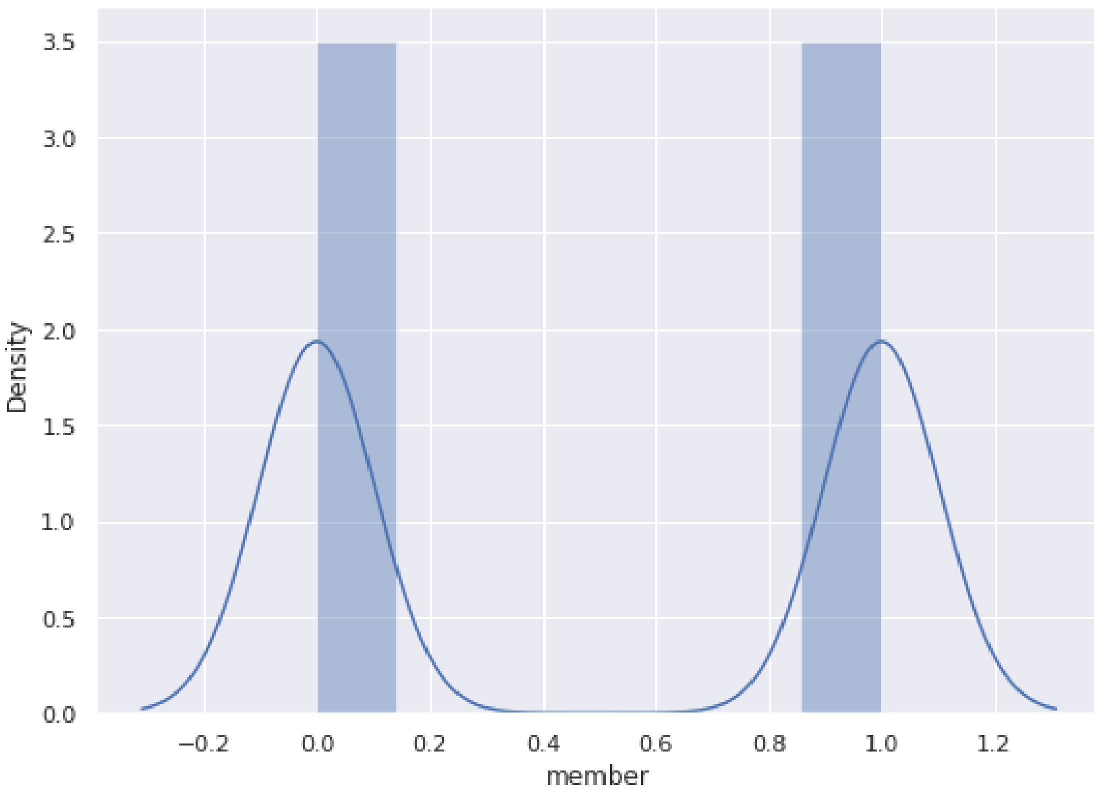
In [398]:
```
targets.value_counts()
```

Out[398]:
```
1    1345
0    1345
Name: member, dtype: int64
```

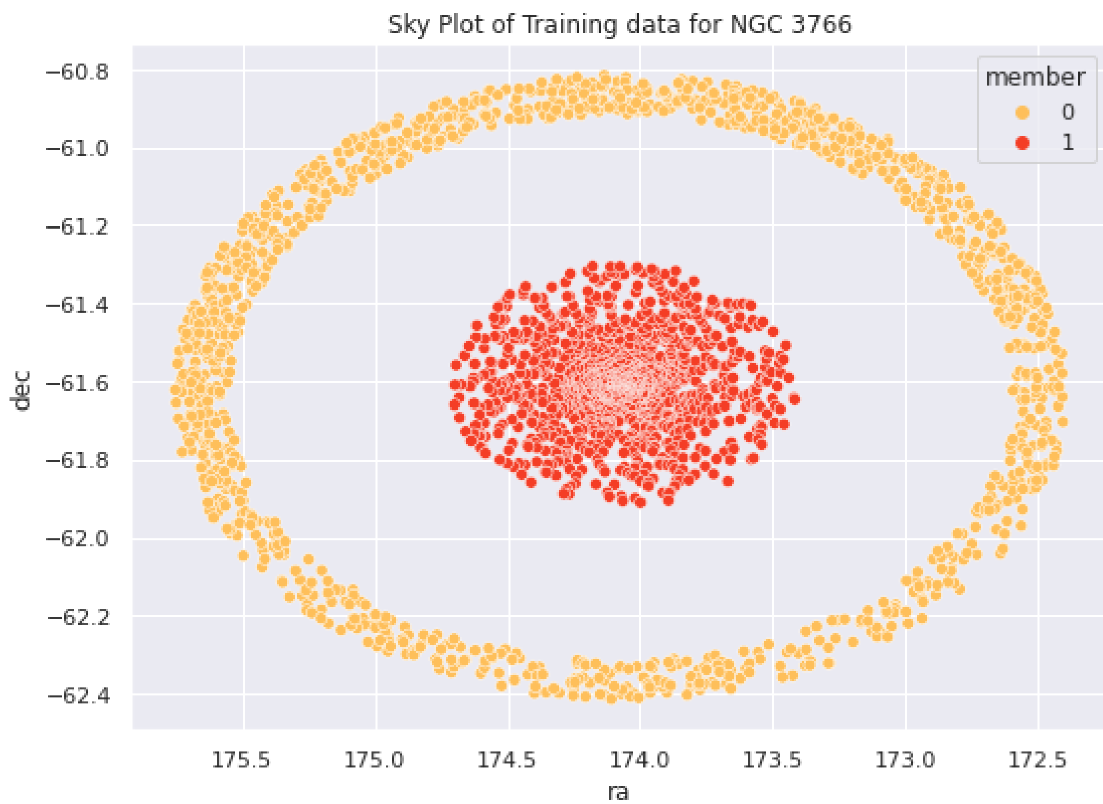
Visualizing Training Data

In [396]:
```
# histogram of PMemb in the training data
sns.distplot(training_set['member'])
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
```

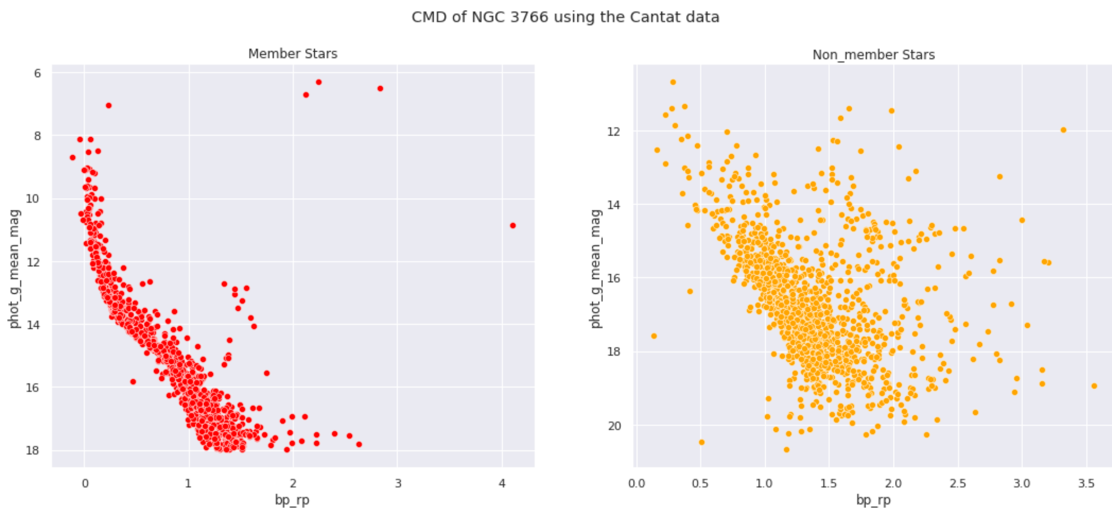


In [399]:
```
sns.set(rc={'figure.figsize':(8.7,6.27)})

skyplot = sns.scatterplot(x = 'ra', y='dec', palette='YlOrRd', hue = 'member', data =
skyplot.invert_xaxis()
plt.title('Sky Plot of Training data for NGC 3766')
plt.show()
```

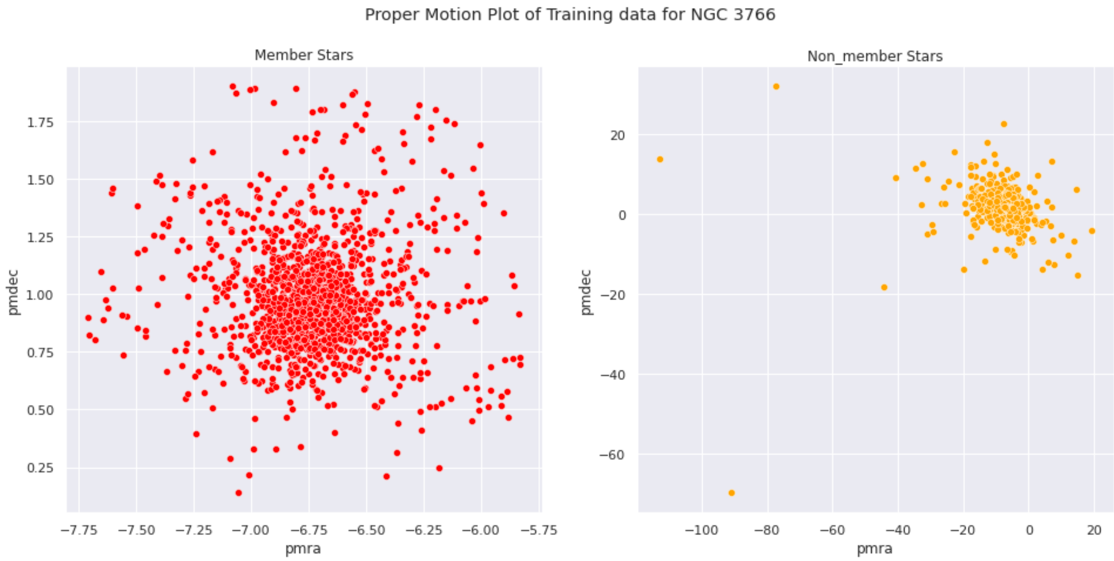



```
In [371]: # CMD marked with the membership probabilities of the stars
          # (PMemb >= 0.5 stars are the probable stars)
          #cmd = sns.scatterplot(x = 'bp_rp', y='phot_g_mean_mag', palette='YlOrRd', hue = 'PM
          #cmd.invert_yaxis()
          #plt.title('')
          #plt.show()

          # proper motion plot marked with the membership probabilities of the stars

          fig, axes = plt.subplots(1, 2, figsize=(18,7))
          fig.suptitle('CMD of Training data for NGC 3766 ')

          sns.scatterplot(x = 'bp_rp', y='phot_g_mean_mag', palette='YlOrRd', color = 'red',
                          data = member, ax = axes[0])
          axes[0].set_title('Member Stars')
          axes[0].invert_yaxis()

          #plt.show()
          sns.scatterplot(x = 'bp_rp', y='phot_g_mean_mag', palette='YlOrRd', color = 'orange'
                          data = non_member, ax = axes[1])
          axes[1].set_title('Non_member Stars')
```

```
axes[1].invert_yaxis()

plt.show()
```



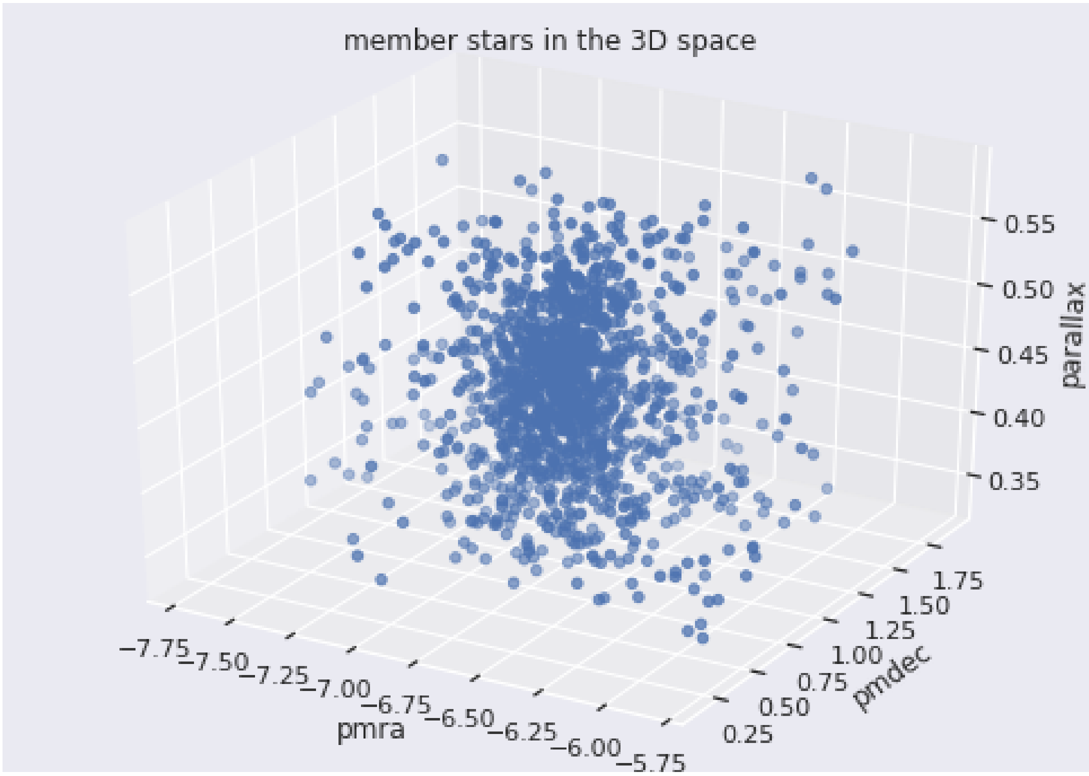
In [561]:
```
# proper motion plot marked with the membership probabilities of the stars

fig, axes = plt.subplots(1, 2, figsize=(16,7))
fig.suptitle('Proper Motion Plot of Training data for NGC 3766 ')

sns.scatterplot(x = 'pmra', y='pmdec', palette='YlOrRd', color = 'red',
                data = member, ax = axes[0])
axes[0].set_title('Member Stars')
#plt.show()
sns.scatterplot(x = 'pmra', y='pmdec', palette='YlOrRd', color = 'orange',
                data = non_member, ax = axes[1])
axes[1].set_title('Non_member Stars')
plt.show()
```



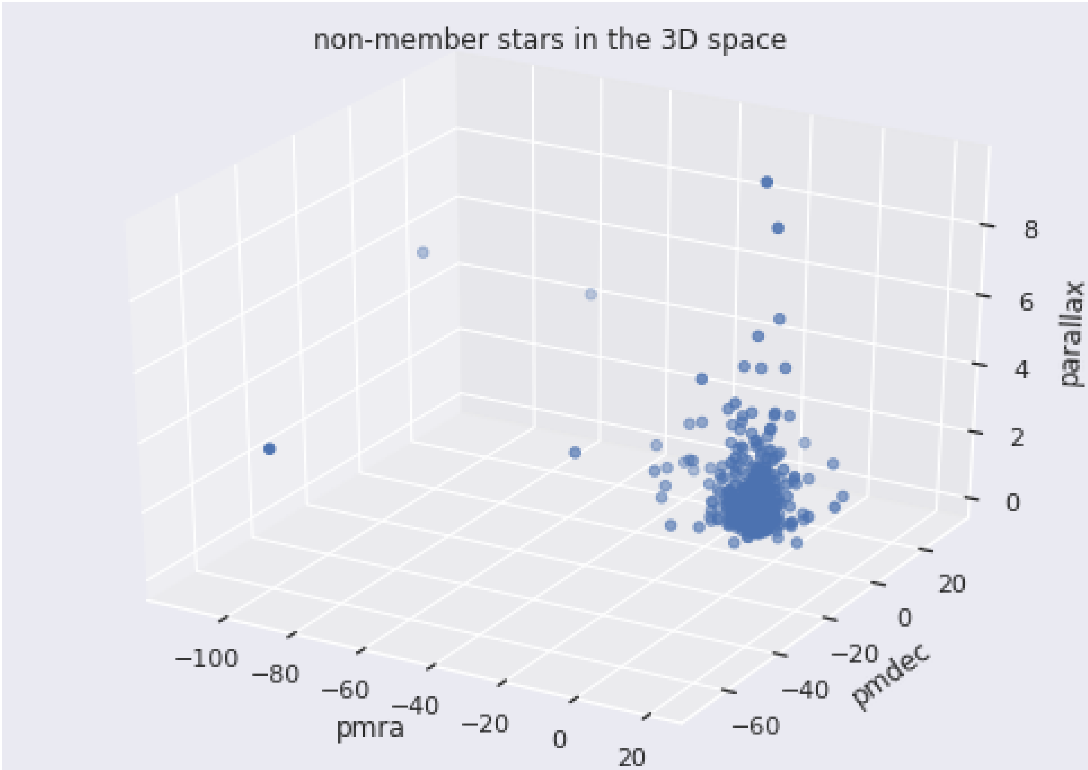

```
In [400]: sns.distplot(member.parallax, label='member')
          sns.distplot(non_member.parallax, label = 'non_member')
          plt.xlim(-0.5,6)
          plt.title('Parallax of Training data for NGC 3766 ')
          plt.legend()
          plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
```

In [401]:
```
from mpl_toolkits.mplot3d import Axes3D


ax = plt.figure().gca(projection='3d')

ax.scatter(member.pmra, member.pmdec, member.parallax)
ax.set_xlabel('pmra')
ax.set_ylabel('pmdec')
ax.set_zlabel('parallax')
plt.title('member stars in the 3D space')
plt.show()
```




```
In [402]: ax = plt.figure().gca(projection='3d')

          ax.scatter(non_member.pmra, non_member.pmdec, non_member.parallax)
          ax.set_xlabel('pmra')
          ax.set_ylabel('pmdec')
          ax.set_zlabel('parallax')
          plt.title('non-member stars in the 3D space')
          plt.show()
```



Check Feature Importance using Random Forest

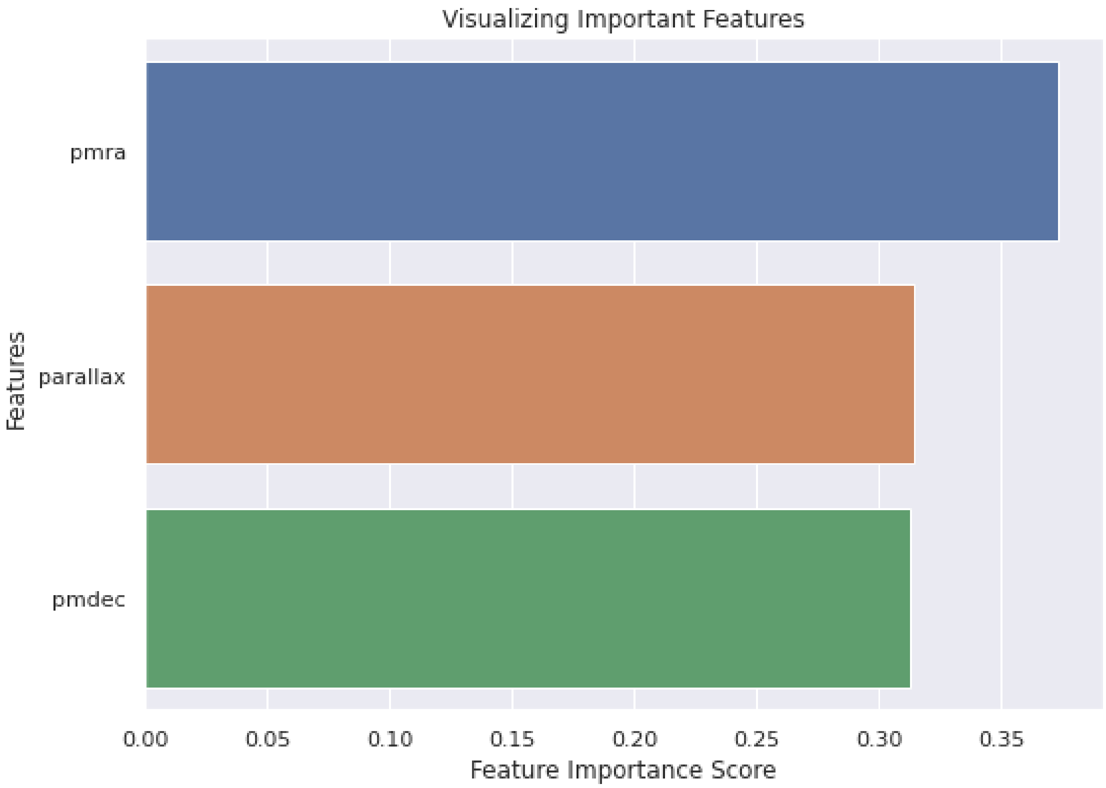
In [403]:
```
# Use Random Forest on whole dataset using 100 different trees
rfc = RandomForestClassifier(n_estimators = 100, oob_score = True)
rfc.fit(features, targets)
```

Out[403]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=True, random_state=None,
                       verbose=0, warm_start=False)
```

In [404]:
```
# checking the feature importance,
# ( this is higher for a variable if the model efficiency become lower as we remove
feature_imp = pd.Series(rfc.feature_importances_, index=features.columns).sort_values
feature_imp
```

Out[404]:
```
pmra        0.373074
parallax    0.314468
pmdec       0.312458
dtype: float64
```

In [405]:
```
# plotting as a barplot

# Creating a bar plot
sns.barplot(x=feature_imp, y=feature_imp.index)
# Add labels to the graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.show()
```



0.0.3 Model Selection and Accuracy Estimate using Test Data

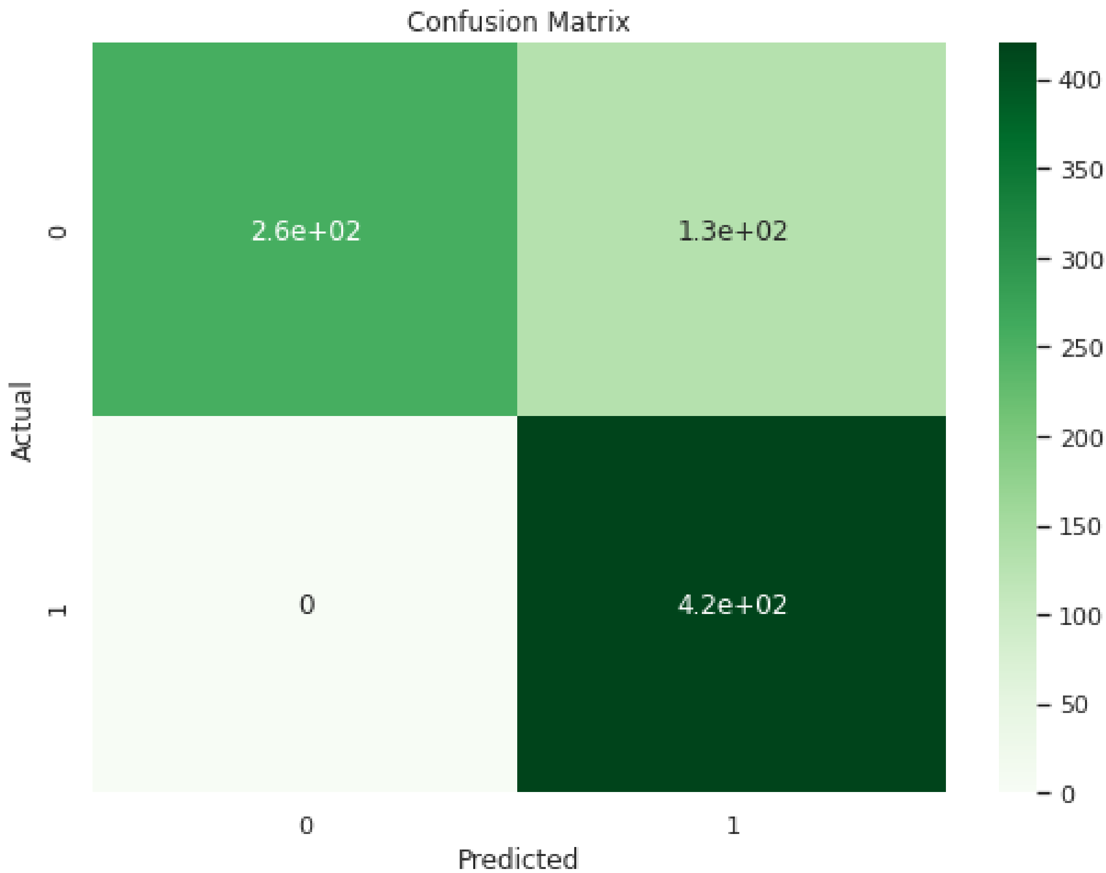
In [441]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.metrics import precision_score, recall_score, classification_report
```

In [551]:
```
from sklearn.model_selection import train_test_split
# splitting our dataset using 0.3 test ratio (30% test data, 70% train data)
train_features, test_features, train_targets, test_targets = train_test_split(feature
                                                                              targe
                                                                              test_
                                                                              rando
```

```
In [443]: def evaluate_model(model):
              test_predict = model.predict(test_features)
              train_predict = model.predict(train_features)

              print('Model Accuracy:')
              print("Precision on training data: %.3f" % precision_score(train_targets, train_p
              print("Precision on testing data: %.3f" % precision_score(test_targets, test_pred
              print('Accuracy on test data: %.3f' % accuracy_score(test_targets, test_predict))


              sns.heatmap(confusion_matrix(test_targets, test_predict), cmap= 'Greens', annot =
              plt.ylabel('Actual')
              plt.xlabel('Predicted')
              plt.title('Confusion Matrix')
              plt.show()

              print("Classification Report: \n", classification_report(test_targets, test_predi
```

SVC

```
In [549]: from sklearn.svm import SVC

          # SVC model
          svc_clf = SVC(kernel='rbf', gamma = 'scale', random_state=42)

          svc_clf.fit(train_features, train_targets)

Out[549]: SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
              decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
              max_iter=-1, probability=False, random_state=42, shrinking=True, tol=0.001,
              verbose=False)

In [550]: evaluate_model(svc_clf)

Model Accuracy:
Precision on training data: 0.751
Precision on testing data: 0.762
Accuracy on test data: 0.838
```

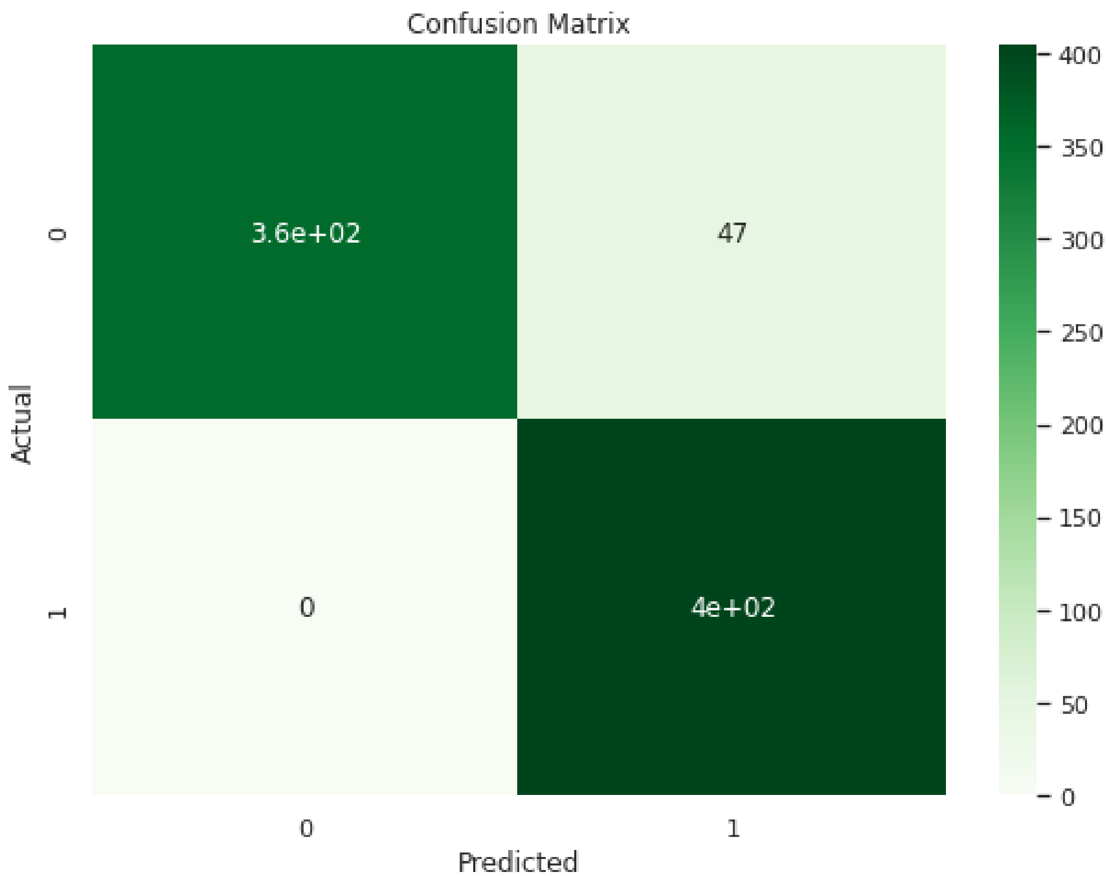



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.66      0.80       387
           1       0.76      1.00      0.87       420

    accuracy                           0.84       807
   macro avg       0.88      0.83      0.83       807
weighted avg       0.88      0.84      0.83       807
```

Naive Bayes

In [459]:
```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(train_features, train_targets)
evaluate_model(gnb)
```

```
Model Accuracy:
Precision on training data: 0.879
Precision on testing data: 0.896
Accuracy on test data: 0.942
```




```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.88      0.94       403
           1       0.90      1.00      0.95       404

    accuracy                           0.94       807
   macro avg       0.95      0.94      0.94       807
weighted avg       0.95      0.94      0.94       807
```

KNN

```
In [446]: from sklearn import neighbors
          from sklearn.model_selection import cross_val_score, GridSearchCV

          knn_cv = neighbors.KNeighborsClassifier()
          parameter_grid = {'n_neighbors': [1,2,3,4,5,6,7,8]}

          #use gridsearch to test all values for n_neighbors
          knn_gscv = GridSearchCV(knn_cv, parameter_grid, cv=5, scoring='precision')

          #fit model to data
          knn_gscv.fit(train_features, train_targets)
```

```
Out[446]: GridSearchCV(cv=5, error_score=nan,
                       estimator=KNeighborsClassifier(algorithm='auto', leaf_size=30,
                                                      metric='minkowski',
                                                      metric_params=None, n_jobs=None,
                                                      n_neighbors=5, p=2,
                                                      weights='uniform'),
                       iid='deprecated', n_jobs=None,
                       param_grid={'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8]},
                       pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                       scoring='precision', verbose=0)
```

```
In [447]: # top performance
          print("Top Performance: ", knn_gscv.best_params_)
          # score for top_performance
          print("Top CV score: ", knn_gscv.best_score_)
```

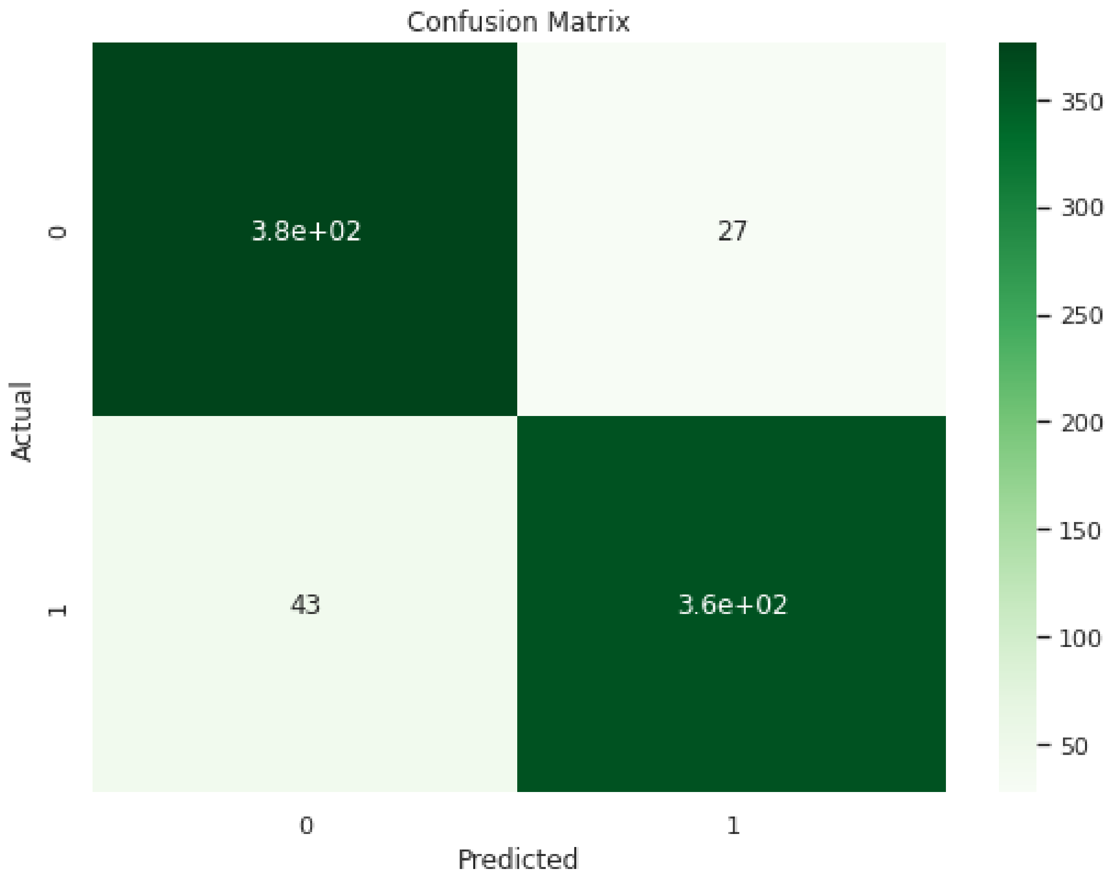
```
Top Performance:  {'n_neighbors': 2}
Top CV score:  0.9211235624022134
```

```
In [448]: n_neighbors = 2
          knn = neighbors.KNeighborsClassifier(n_neighbors,)
          knn.fit(train_features, train_targets)
```

```
Out[448]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                               metric_params=None, n_jobs=None, n_neighbors=2, p=2,
                               weights='uniform')
```

```
In [449]: evaluate_model(knn)
```

```
Model Accuracy:
Precision on training data: 1.000
Precision on testing data: 0.930
Accuracy on test data: 0.913
```




```
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.93      0.91       403
           1       0.93      0.89      0.91       404

    accuracy                           0.91       807
   macro avg       0.91      0.91      0.91       807
weighted avg       0.91      0.91      0.91       807
```

Decision Tree

```
In [450]: from sklearn import tree
          dtc = tree.DecisionTreeClassifier()
          dtc.fit(train_features, train_targets)
          test_predict = dtc.predict(test_features)

In [451]: dtc.get_params()
```

```
Out[451]: {'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'presort': 'deprecated',
 'random_state': None,
 'splitter': 'best'}
```

```
In [452]: from sklearn.model_selection import RandomizedSearchCV

          max_features = ['auto', 'sqrt']
          # Maximum number of levels
          max_depth = [int(x) for x in np.linspace(10, 100, num = 10)]
          max_depth.append(None)
          min_samples_split = [2, 5, 10]
          min_samples_leaf = [1, 2, 4]
          np.random.seed(25)
          random_states = np.random.choice(range(1,50), size = 10, replace=False)
          ccp_alpha = [2**i for i in range(-10,0)]

          random_grid = {'max_features': max_features,
                         'max_depth': max_depth,
                         'min_samples_split': min_samples_split,
                         'min_samples_leaf': min_samples_leaf,
                         'random_state' : random_states,
                         'ccp_alpha': ccp_alpha}

          random_grid
```

```
Out[452]: {'ccp_alpha': [0.0009765625,
  0.001953125,
  0.00390625,
  0.0078125,
  0.015625,
  0.03125,
  0.0625,
  0.125,
  0.25,
  0.5],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, None],
 'max_features': ['auto', 'sqrt'],
```

```
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'random_state': array([44, 15, 41, 34, 48,  8, 33, 20, 39, 36])}
```

In [453]:
```python
# base model
dtc = tree.DecisionTreeClassifier()

dtc_random = RandomizedSearchCV(estimator = dtc, param_distributions = random_grid,
                               n_iter = 100, cv = 5, verbose=2, random_state=42, n_jo
                               scoring = 'precision')
```

In [454]:
```python
dtc_random.fit(train_features, train_targets)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done 186 tasks      | elapsed:    2.7s
[Parallel(n_jobs=-1)]: Done 500 out of 500 | elapsed:    4.5s finished
```

Out[454]:
```
RandomizedSearchCV(cv=5, error_score=nan,
                   estimator=DecisionTreeClassifier(ccp_alpha=0.0,
                                                    class_weight=None,
                                                    criterion='gini',
                                                    max_depth=None,
                                                    max_features=None,
                                                    max_leaf_nodes=None,
                                                    min_impurity_decrease=0.0,
                                                    min_impurity_split=None,
                                                    min_samples_leaf=1,
                                                    min_samples_split=2,
                                                    min_weight_fraction_leaf=0.0,
                                                    presort='deprecated',
                                                    random_state=None,
                                                    splitter='best'),
                   i...
                                                      0.00390625, 0.0078125,
                                                      0.015625, 0.03125, 0.0625,
                                                      0.125, 0.25, 0.5],
                                        'max_depth': [10, 20, 30, 40, 50, 60,
                                                      70, 80, 90, 100, None],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 4],
                                        'min_samples_split': [2, 5, 10],
                                        'random_state': array([44, 15, 41, 34, 48,  8
                   pre_dispatch='2*n_jobs', random_state=42, refit=True,
                   return_train_score=False, scoring='precision', verbose=2)
```

```
In [455]:  dtc_random.best_params_
```

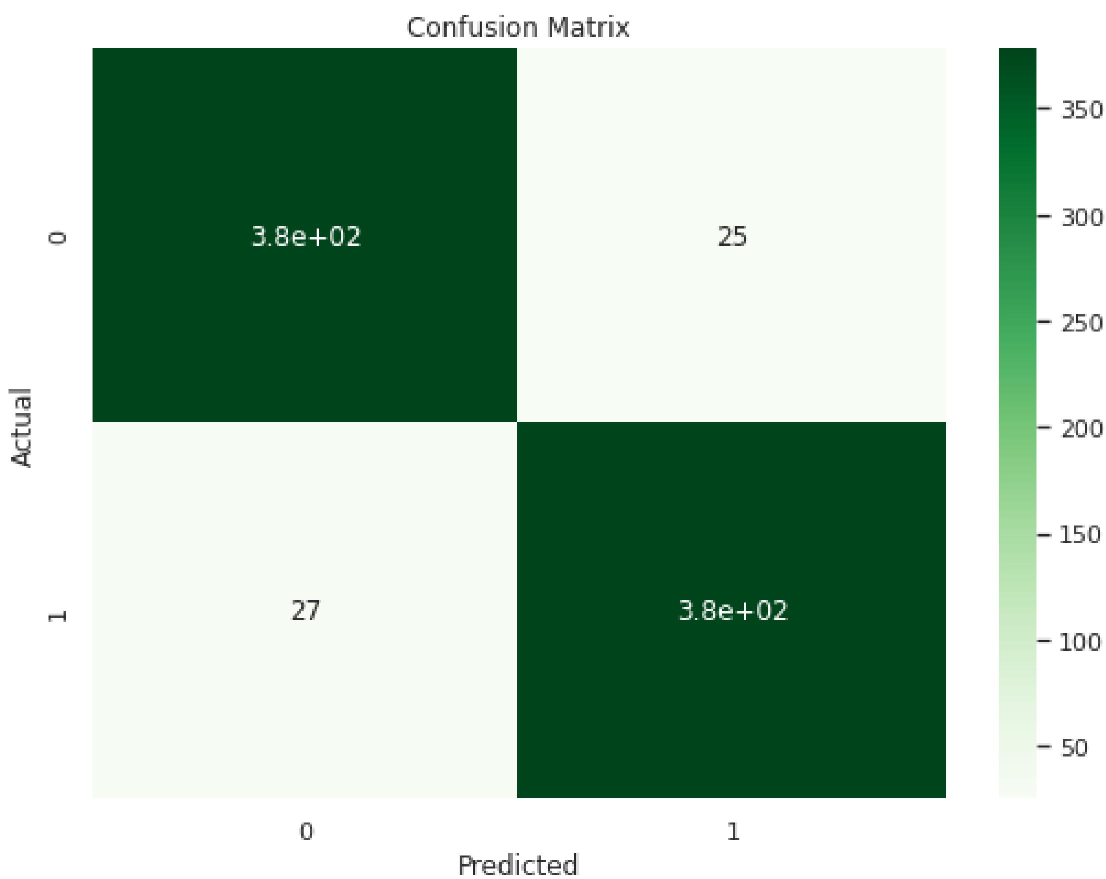
```
Out[455]: {'ccp_alpha': 0.0078125,
           'max_depth': 20,
           'max_features': 'auto',
           'min_samples_leaf': 4,
           'min_samples_split': 2,
           'random_state': 48}
```

```
In [456]:  dtc_random.best_score_
```

```
Out[456]: 0.9369494618862596
```

```
In [457]: base_model_dtc = tree.DecisionTreeClassifier()
          base_model_dtc.fit(train_features, train_targets)

          evaluate_model(base_model_dtc)
```

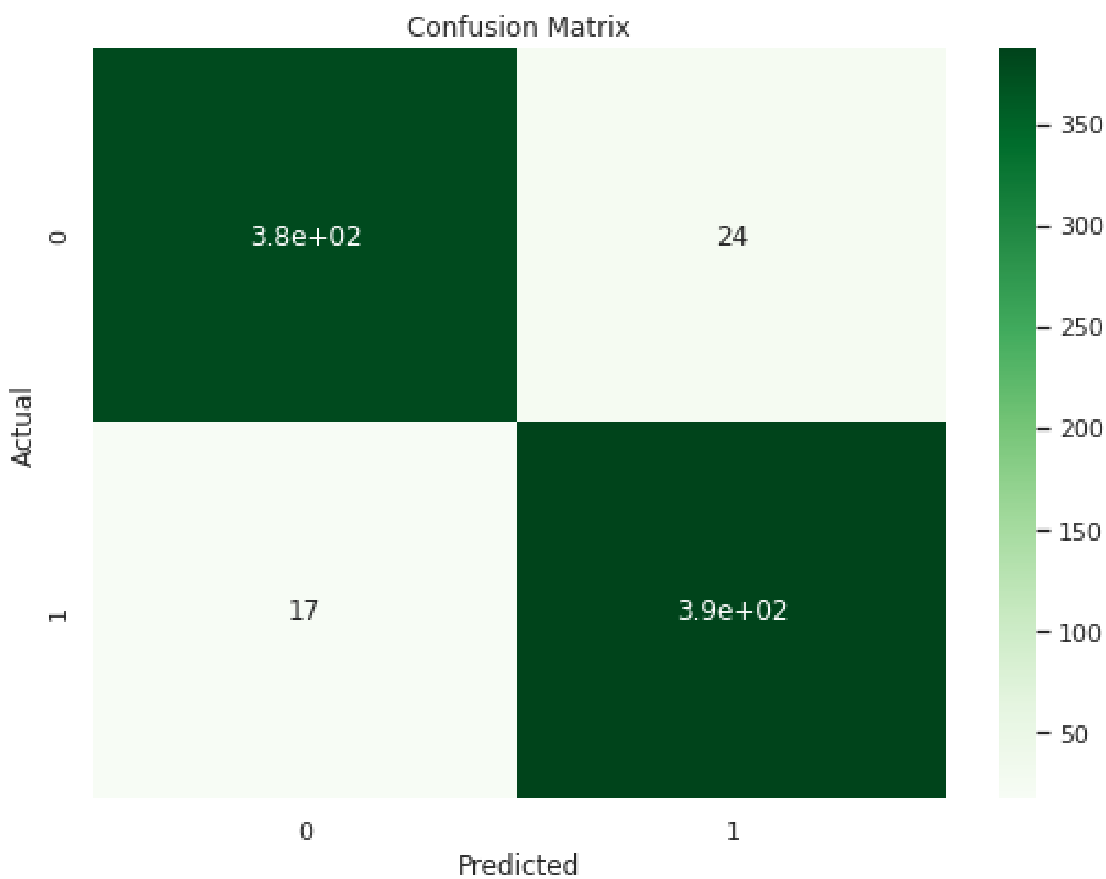
```
Model Accuracy:
Precision on training data: 1.000
Precision on testing data: 0.938
Accuracy on test data: 0.936
```



```
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.94      0.94       403
           1       0.94      0.93      0.94       404

    accuracy                           0.94       807
   macro avg       0.94      0.94      0.94       807
weighted avg       0.94      0.94      0.94       807
```

```
In [458]: best_random_dtc = dtc_random.best_estimator_

          evaluate_model(best_random_dtc)
```

```
Model Accuracy:
Precision on training data: 0.937
Precision on testing data: 0.942
Accuracy on test data: 0.949
```



```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95       403
           1       0.94      0.96      0.95       404

    accuracy                           0.95       807
   macro avg       0.95      0.95      0.95       807
weighted avg       0.95      0.95      0.95       807
```

Random Forest

In [465]:
```
from sklearn.model_selection import RandomizedSearchCV

n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1000, num = 10)]
max_features = ['auto', 'sqrt']
# Maximum number of levels
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)
min_samples_split = [2, 5, 10]
min_samples_leaf = [1, 2, 4]
bootstrap = [True, False]
ccp_alpha = [2**i for i in range(-10,0)]

random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap,
               'ccp_alpha': ccp_alpha}

random_grid
```

Out[465]:
```
{'bootstrap': [True, False],
 'ccp_alpha': [0.0009765625,
  0.001953125,
  0.00390625,
  0.0078125,
  0.015625,
  0.03125,
  0.0625,
  0.125,
```

```
             0.25,
             0.5],
            'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
            'max_features': ['auto', 'sqrt'],
            'min_samples_leaf': [1, 2, 4],
            'min_samples_split': [2, 5, 10],
            'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]}
```

```
In [466]: rfc = RandomForestClassifier()

          rfc_random = RandomizedSearchCV(estimator = rfc, param_distributions = random_grid,
                                  n_iter = 100, cv = 5, verbose=2, random_state=42, n_jo
                                  scoring = 'precision')
```

```
In [467]: rfc_random.fit(train_features, train_targets)
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits


[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:   37.9s
[Parallel(n_jobs=-1)]: Done 158 tasks      | elapsed:  3.3min
[Parallel(n_jobs=-1)]: Done 361 tasks      | elapsed:  6.9min
[Parallel(n_jobs=-1)]: Done 500 out of 500 | elapsed:  9.8min finished
```

```
Out[467]: RandomizedSearchCV(cv=5, error_score=nan,
                             estimator=RandomForestClassifier(bootstrap=True,
                                                              ccp_alpha=0.0,
                                                              class_weight=None,
                                                              criterion='gini',
                                                              max_depth=None,
                                                              max_features='auto',
                                                              max_leaf_nodes=None,
                                                              max_samples=None,
                                                              min_impurity_decrease=0.0,
                                                              min_impurity_split=None,
                                                              min_samples_leaf=1,
                                                              min_samples_split=2,
                                                              min_weight_fraction_leaf=0.0,
                                                              n_estimators=100,
                                                              n_jobs...
                                                                0.00390625, 0.0078125,
                                                                0.015625, 0.03125, 0.0625,
                                                                0.125, 0.25, 0.5],
                                                  'max_depth': [10, 20, 30, 40, 50, 60,
                                                                70, 80, 90, 100, 110,
                                                                None],
                                                  'max_features': ['auto', 'sqrt'],
```

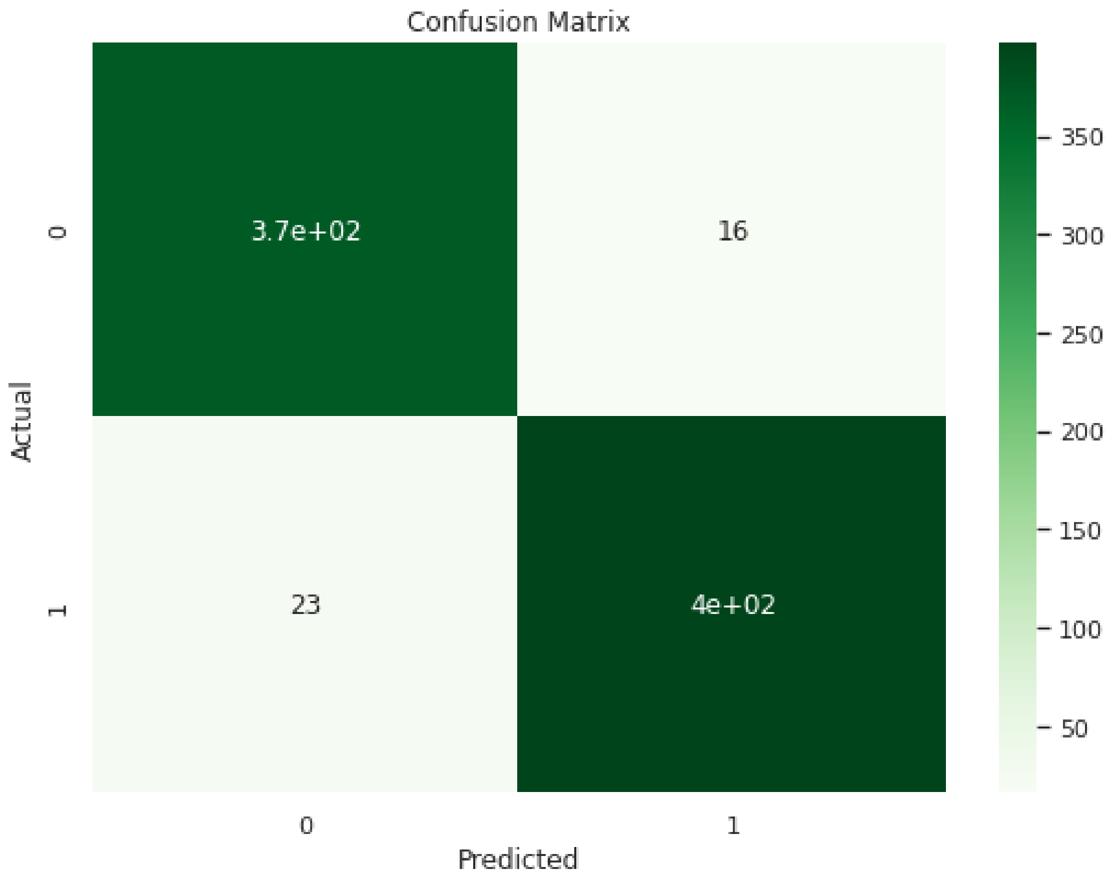
```
                                        'min_samples_leaf': [1, 2, 4],
                                        'min_samples_split': [2, 5, 10],
                                        'n_estimators': [100, 200, 300, 400,
                                                         500, 600, 700, 800,
                                                         900, 1000]},
                   pre_dispatch='2*n_jobs', random_state=42, refit=True,
                   return_train_score=False, scoring='precision', verbose=2)
```

In [468]:
```
rfc_random.best_params_
```

Out[468]:
```
{'bootstrap': True,
 'ccp_alpha': 0.0078125,
 'max_depth': None,
 'max_features': 'sqrt',
 'min_samples_leaf': 1,
 'min_samples_split': 10,
 'n_estimators': 200}
```

In [470]:
```
base_model = RandomForestClassifier(n_estimators = 100, random_state = 42,
                                    oob_score = True)
base_model.fit(train_features, train_targets)

evaluate_model(base_model)
```

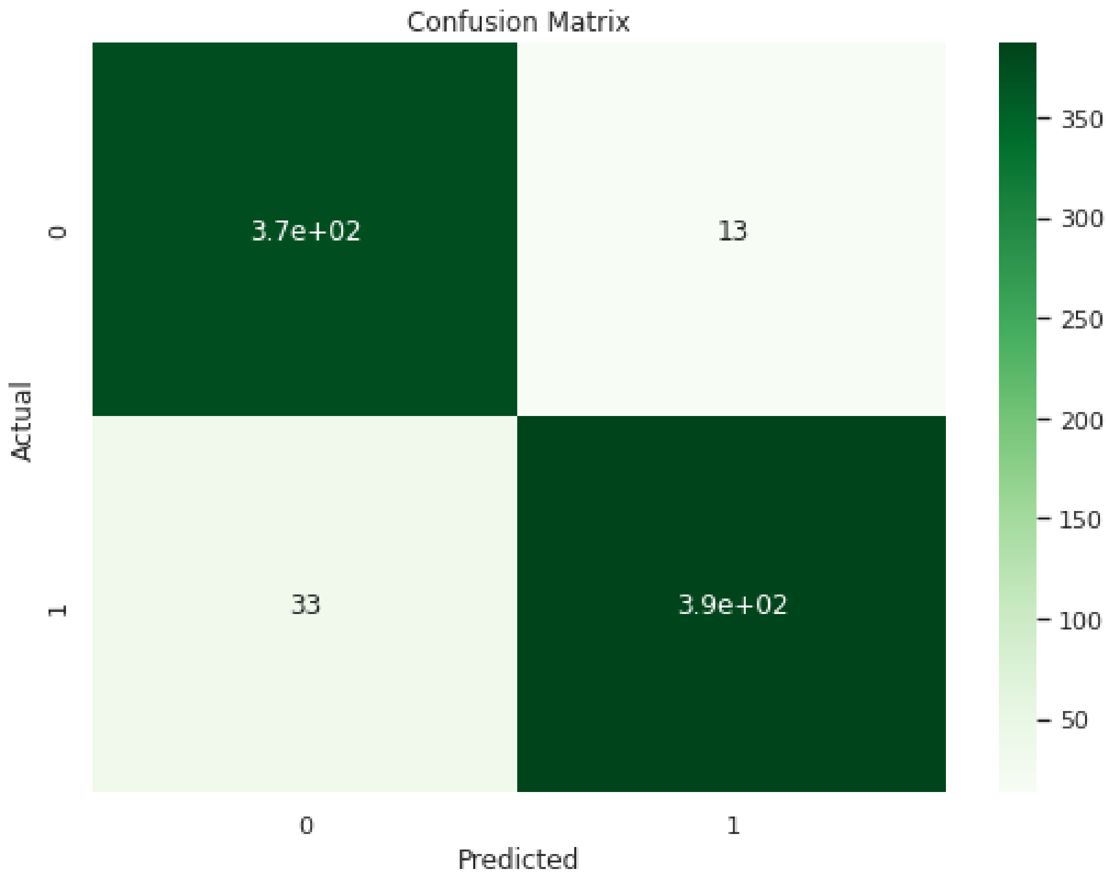
```
Model Accuracy:
Precision on training data: 1.000
Precision on testing data: 0.961
Accuracy on test data: 0.952
```




```
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.96      0.95       387
           1       0.96      0.95      0.95       420

    accuracy                           0.95       807
   macro avg       0.95      0.95      0.95       807
weighted avg       0.95      0.95      0.95       807
```

In [471]:
```
best_random = rfc_random.best_estimator_

evaluate_model(best_random)
```

```
Model Accuracy:
Precision on training data: 0.946
Precision on testing data: 0.968
Accuracy on test data: 0.943
```




```
Classification Report:
              precision    recall  f1-score   support

           0       0.92      0.97      0.94       387
           1       0.97      0.92      0.94       420

    accuracy                           0.94       807
   macro avg       0.94      0.94      0.94       807
weighted avg       0.94      0.94      0.94       807
```

0.0.4 Prediction for the new stars

In [472]:
```
# descriptive stats
all_stars_filtered.describe()
```

Out[472]:
```
        solution_id     source_id  ...  pmdec_over_error  dist_3766_center
count  8.849200e+04  8.849200e+04  ...      88492.000000      88492.000000
mean   1.635721e+18  5.334944e+18  ...         26.082106          0.527082
```

```
std    0.000000e+00  7.299506e+14  ...      41.975123        0.192407
min    1.635721e+18  5.333378e+18  ...       3.000049        0.001113
25%    1.635721e+18  5.334198e+18  ...       8.575895        0.392651
50%    1.635721e+18  5.334899e+18  ...      16.163669        0.561294
75%    1.635721e+18  5.335699e+18  ...      29.967447        0.687794
max    1.635721e+18  5.335930e+18  ...    3562.870471        0.803857

[8 rows x 94 columns]
```

In [473]:
```python
# chosing only GAIA stars close to 0.40 degree radius of the center
all_stars_filtered = all_stars_filtered.dropna(subset = feature_columns)
GAIA_target_stars = all_stars_filtered[all_stars_filtered['dist_3766_center'] <= 0.40
```

In [475]:
```python
# removing the member stars from GAIA data
GAIA_target_stars = pd.concat([GAIA_target_stars, training_data.drop(columns=['PMemb
                              training_data.drop(columns=['PMemb', 'member'])]).drop
```

In [476]:
```python
# select the set of predictor variables from the new dataset
new_features = GAIA_target_stars.loc[:, feature_columns]
new_features = new_features.astype('float32')
```

In [477]:
```python
# train the model again using all the features and targets of the previous dataset
# rfc.fit(features, targets)
```

In [478]:
```python
# estimate the membership classification of the stars
GAIA_target_stars['member'] = best_random.predict(new_features)
GAIA_target_stars['member'].value_counts()
```

Out[478]:
```
0    20746
1      828
Name: member, dtype: int64
```

In [479]:
```python
# estimate the membership probability of the stars
GAIA_target_stars['PMemb'] = best_random.predict_proba(new_features)[:,1]
sum(GAIA_target_stars['PMemb'] >= 0.5)
```

Out[479]: 828

In [552]:
```python
potentialMember = GAIA_target_stars[GAIA_target_stars['member'] == 1]
len(potentialMember)
```

Out[552]: 828

In [485]:
```python
potentialMember.describe()
```

Out[485]:
```
        solution_id     source_id  ...  member       PMemb
count  8.280000e+02  8.280000e+02  ...   828.0  828.000000
mean   1.635721e+18  5.334879e+18  ...     1.0    0.777271
std    0.000000e+00  7.489140e+14  ...     0.0    0.137153
min    1.635721e+18  5.334126e+18  ...     1.0    0.502270
```

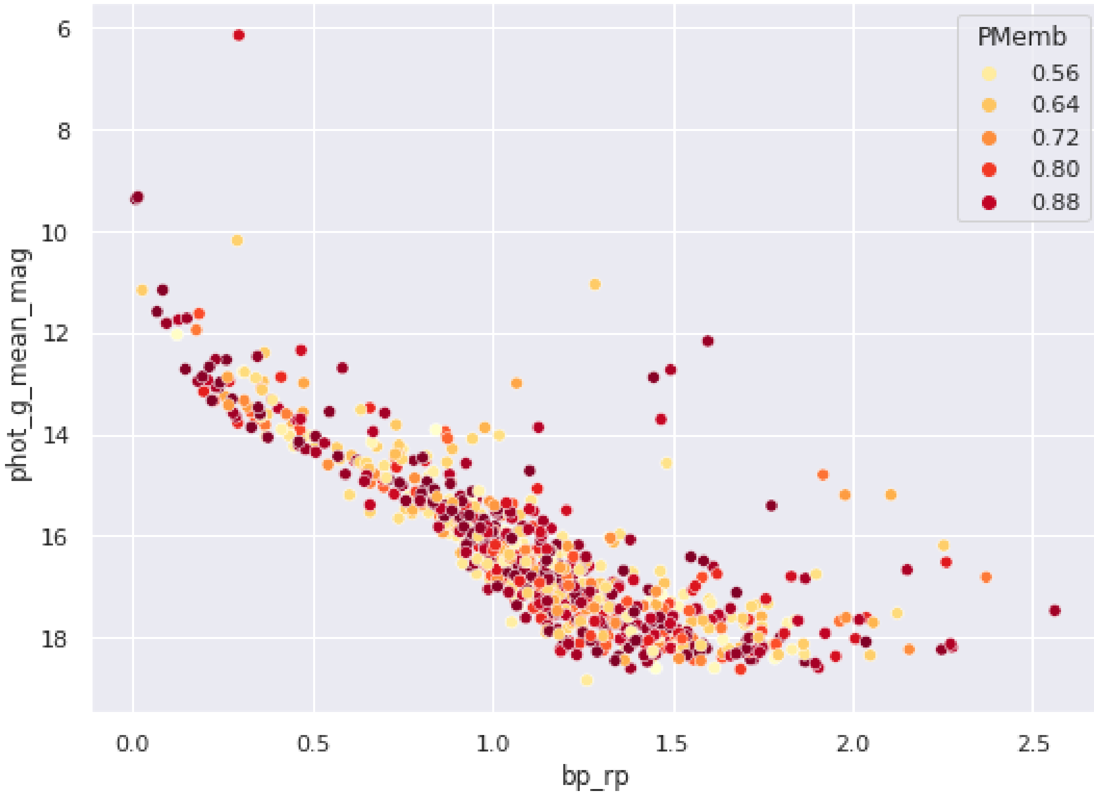
```
25%    1.635721e+18  5.334178e+18  ...      1.0    0.633693
50%    1.635721e+18  5.334225e+18  ...      1.0    0.796437
75%    1.635721e+18  5.335670e+18  ...      1.0    0.907752
max    1.635721e+18  5.335728e+18  ...      1.0    0.941000

[8 rows x 96 columns]
```

In [486]:
```
member.describe()
```

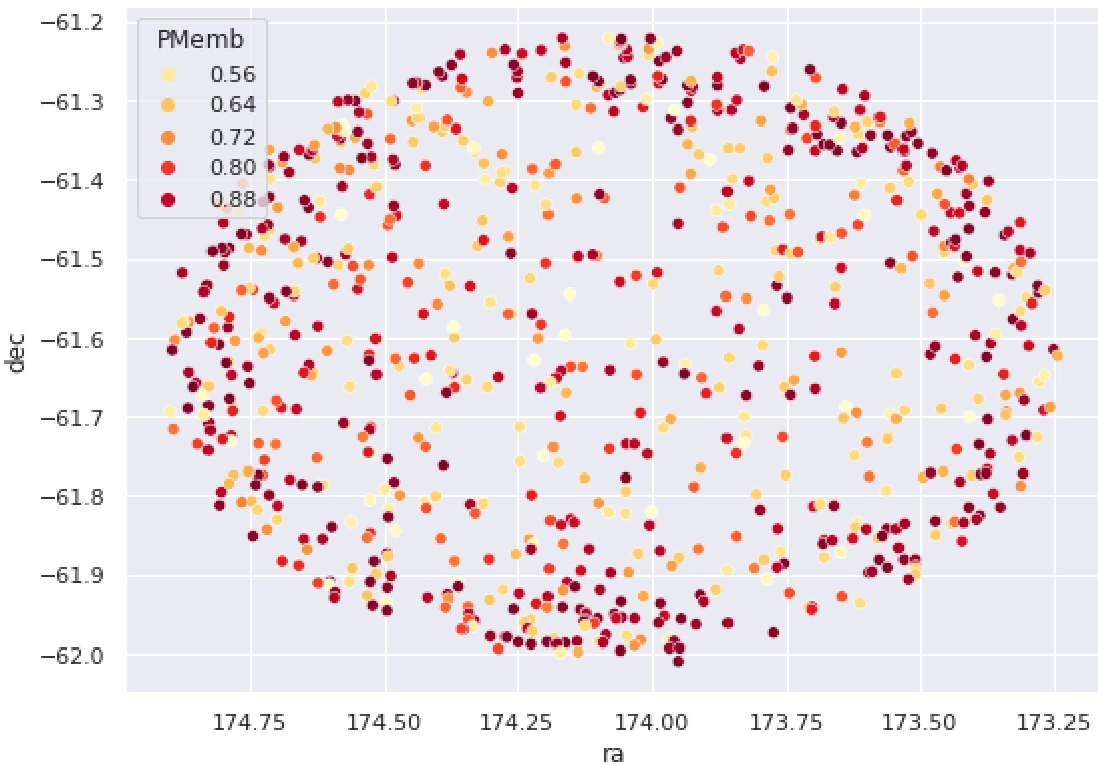
Out[486]:
```
        solution_id     source_id  ...  dist_3766_center  member
count  1.345000e+03  1.345000e+03  ...       1345.000000  1345.0
mean   1.635721e+18  5.334609e+18  ...          0.139332     1.0
std    0.000000e+00  6.676445e+14  ...          0.084147     0.0
min    1.635721e+18  5.334149e+18  ...          0.003027     1.0
25%    1.635721e+18  5.334201e+18  ...          0.068670     1.0
50%    1.635721e+18  5.334209e+18  ...          0.123610     1.0
75%    1.635721e+18  5.335660e+18  ...          0.208140     1.0
max    1.635721e+18  5.335720e+18  ...          0.316216     1.0

[8 rows x 96 columns]
```

Visualization

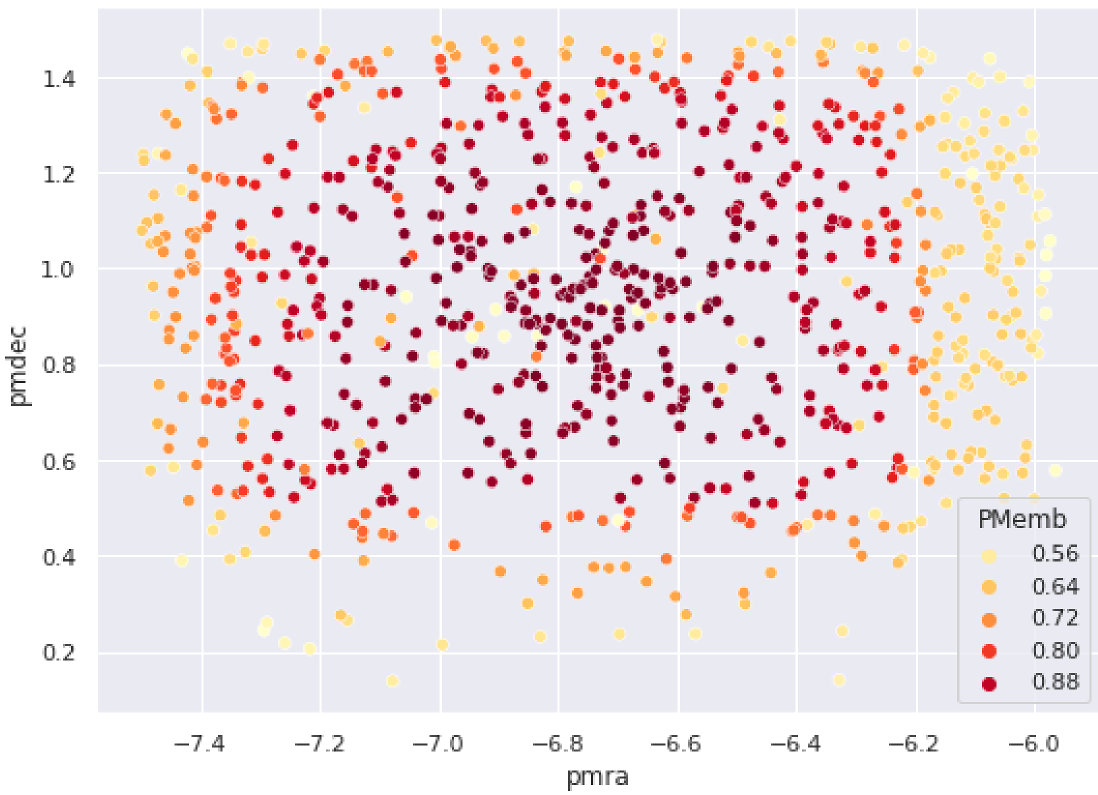
In [480]:
```
# CMD of predicted members

cmd = sns.scatterplot(x = 'bp_rp', y = 'phot_g_mean_mag', hue= 'PMemb',
                      palette='YlOrRd', data = GAIA_target_stars[GAIA_target_stars['I
cmd.invert_yaxis()
```




```
In [481]: skyplot = sns.scatterplot(x = 'ra', y = 'dec', hue= 'PMemb',
                                    palette='YlOrRd', data = GAIA_target_stars[GAIA_target_stars['
          skyplot.invert_xaxis()
```

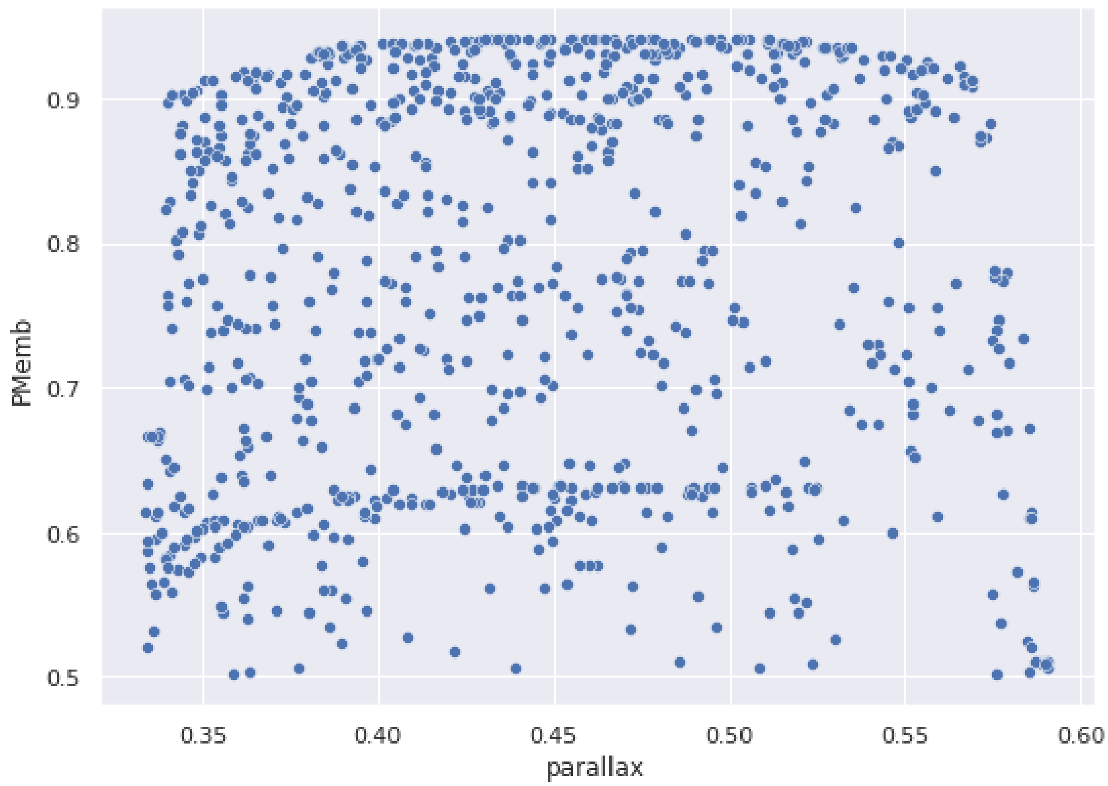



```
In [482]: # pm plot
          sns.scatterplot(x = 'pmra', y = 'pmdec', hue= 'PMemb',
                                         palette='YlOrRd', data = GAIA_target_stars[GAIA_target_stars['I

Out[482]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4e646c8b38>
```




```
In [483]: # parallax plot
          sns.scatterplot(x = 'parallax', y = 'PMemb',
                                      palette='YlOrRd', data = GAIA_target_stars[GAIA_target_stars['I
```

```
Out[483]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4e64659b00>
```




```
In [490]: # saving the files as csv
          # all_stars.to_csv('gaia_3766_membership_prob.csv')

          potentialMember.to_csv('NGC_3766_membership_prob.csv')

In [490]:
```

0.0.5 Comparing the old and new predicted members

```
In [491]: # creating subset for the potential member in previous dataset

          concatenated = pd.concat([potentialMember.assign(dataset='New_member'), member.assign

In [492]: concatenated

Out[492]:                solution_id                    designation  ...     PMemb     dataset
          338     1635721458409799680  Gaia DR2 5334208407957638272  ...  0.563154  New_member
          766     1635721458409799680  Gaia DR2 5334206689970688768  ...  0.908125  New_member
          1322    1635721458409799680  Gaia DR2 5334208133079719040  ...  0.739011  New_member
          1694    1635721458409799680  Gaia DR2 5334206930488832000  ...  0.773206  New_member
          1950    1635721458409799680  Gaia DR2 5334209198231611136  ...  0.632840  New_member
          ...                     ...                            ...  ...       ...         ...
          104667  1635721458409799680  Gaia DR2 5334226240663383808  ...  1.000000  Old_member
```

```
104808  1635721458409799680  Gaia DR2 5335719411837020032  ...  1.000000  Old_member
104811  1635721458409799680  Gaia DR2 5334215623503357056  ...  1.000000  Old_member
104889  1635721458409799680  Gaia DR2 5335716869181854336  ...  1.000000  Old_member
105689  1635721458409799680  Gaia DR2 5334223869841146496  ...  1.000000  Old_member

[2173 rows x 103 columns]
```

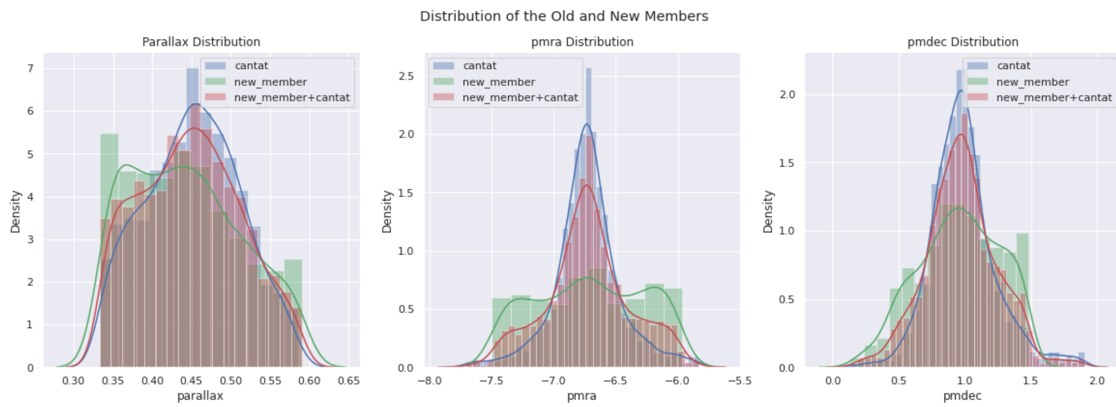
In [493]:
```
concatenated.dataset.value_counts()
```

Out[493]:
```
Old_member    1345
New_member     828
Name: dataset, dtype: int64
```

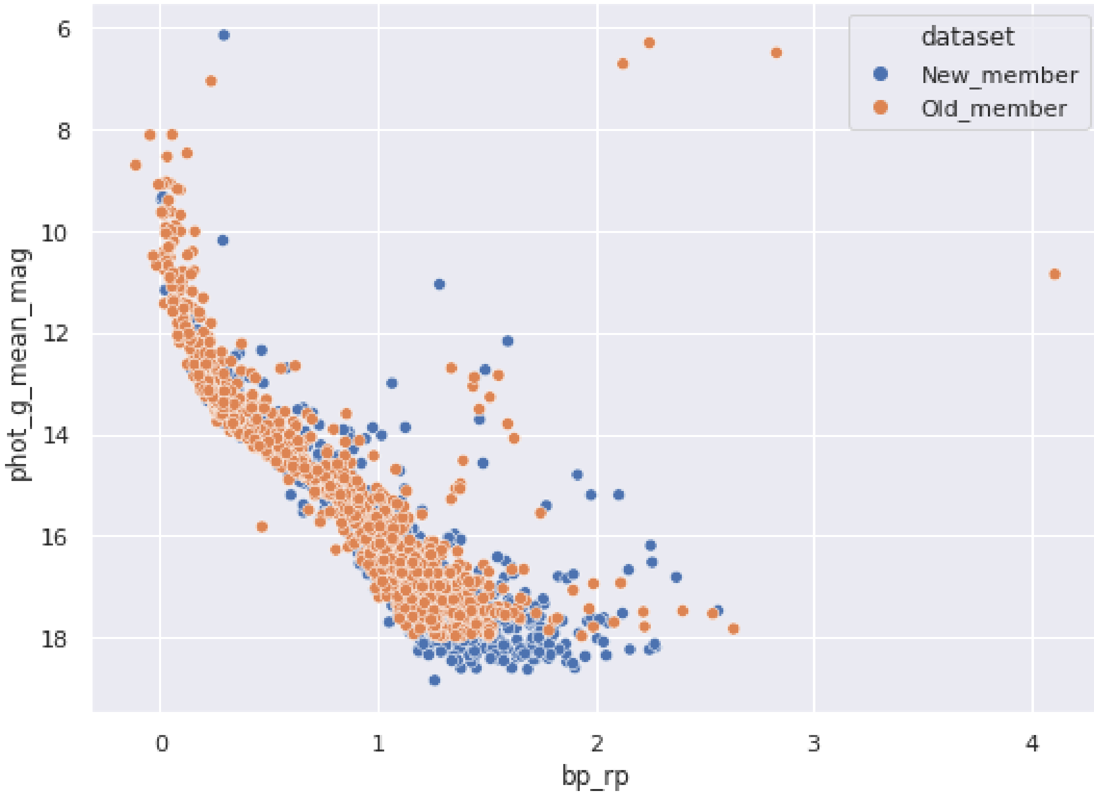
In [560]:
```
fig, axes = plt.subplots(1, 3, figsize=(20,6))
fig.suptitle('Distribution of the Old and New Members')


sns.distplot(member['parallax'], color = 'b', label = 'cantat',
             kde=True, ax=axes[0])
sns.distplot(potentialMember['parallax'], color = 'g', label = 'new_member',
             kde=True, ax=axes[0])
sns.distplot(concatenated['parallax'], color = 'r', ax=axes[0], kde=True,
             label = 'new_member+cantat')
axes[0].set_title('Parallax Distribution')
axes[0].legend()

sns.distplot(member['pmra'], color = 'b', label = 'cantat',
             kde=True, ax=axes[1])
sns.distplot(potentialMember['pmra'], color = 'g', label = 'new_member',
             kde=True, ax=axes[1])
sns.distplot(concatenated['pmra'], color = 'r', ax=axes[1], kde=True,
             label = 'new_member+cantat')
axes[1].set_title('pmra Distribution')
axes[1].legend()

sns.distplot(member['pmdec'], color = 'b', label = 'cantat',
             kde=True, ax=axes[2])
sns.distplot(potentialMember['pmdec'], color = 'g', label = 'new_member',
             kde=True, ax=axes[2])
sns.distplot(concatenated['pmdec'], color = 'r', ax=axes[2], kde=True,
             label = 'new_member+cantat')
axes[2].set_title('pmdec Distribution')
axes[2].legend()

plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot
```

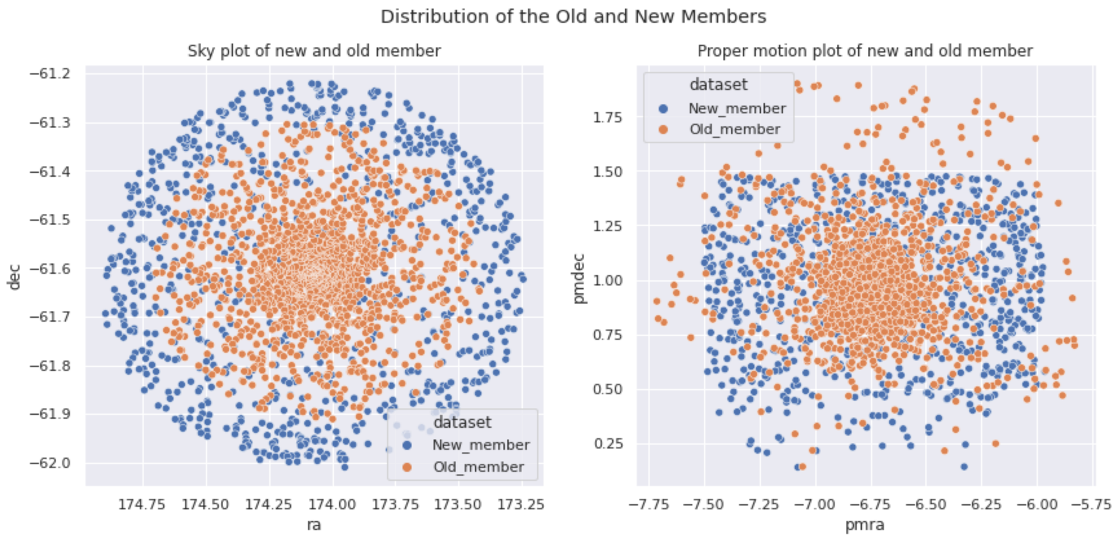
```
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot`
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot`
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot`
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot`
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot`
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot`
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot`
  warnings.warn(msg, FutureWarning)
```



In [494]:
```python
cmd = sns.scatterplot(x='bp_rp', y='phot_g_mean_mag', data=concatenated,
                      hue='dataset')
cmd.invert_yaxis()
```

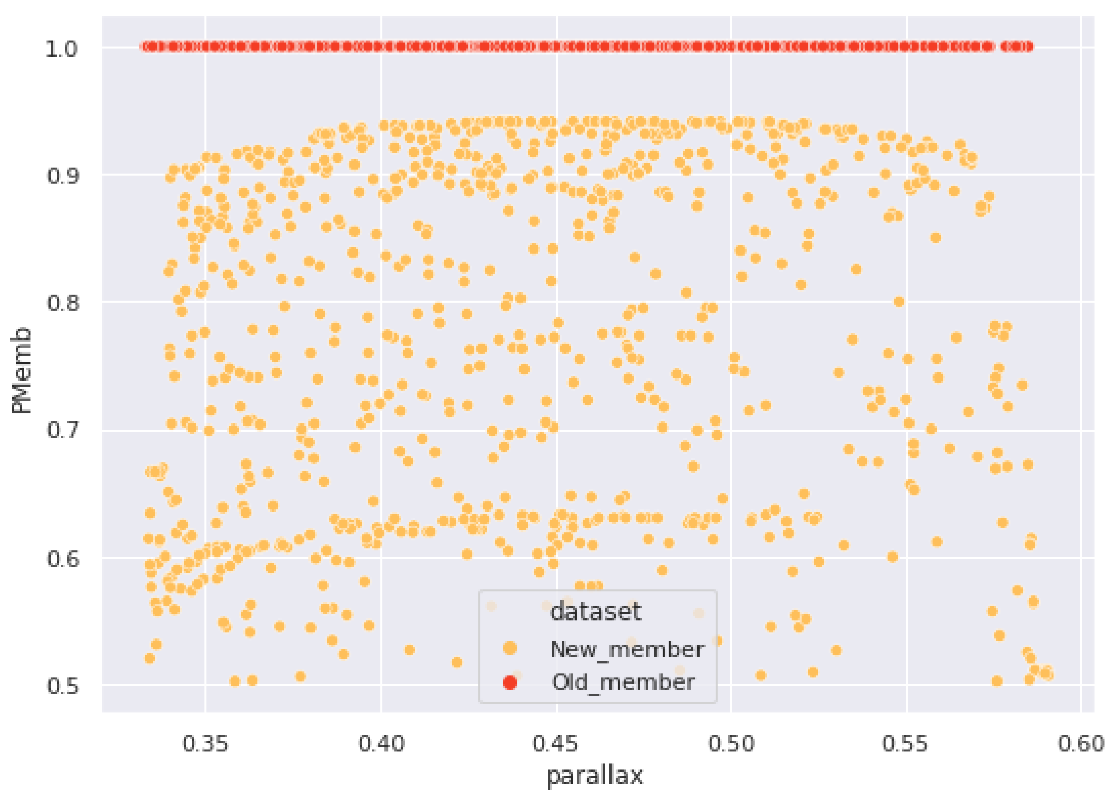



```
In [558]: fig, axes = plt.subplots(1, 2, figsize=(14,6))
          fig.suptitle('Distribution of the Old and New Members')

          skyplot = sns.scatterplot(x='ra', y='dec', data=concatenated,
                                    hue='dataset', ax=axes[0])
          skyplot.invert_xaxis()
          axes[0].set_title('Sky plot of new and old member')

          # proper motion plot
          sns.scatterplot(x='pmra', y='pmdec', data=concatenated,
                                    hue='dataset', ax=axes[1])
          axes[1].set_title('Proper motion plot of new and old member')

          plt.show()
```




```
In [498]: # parallax vs PMemb plot
          sns.scatterplot(x = 'parallax', y = 'PMemb',  hue = 'dataset',
                                          palette='YlOrRd', data = concatenated )
```

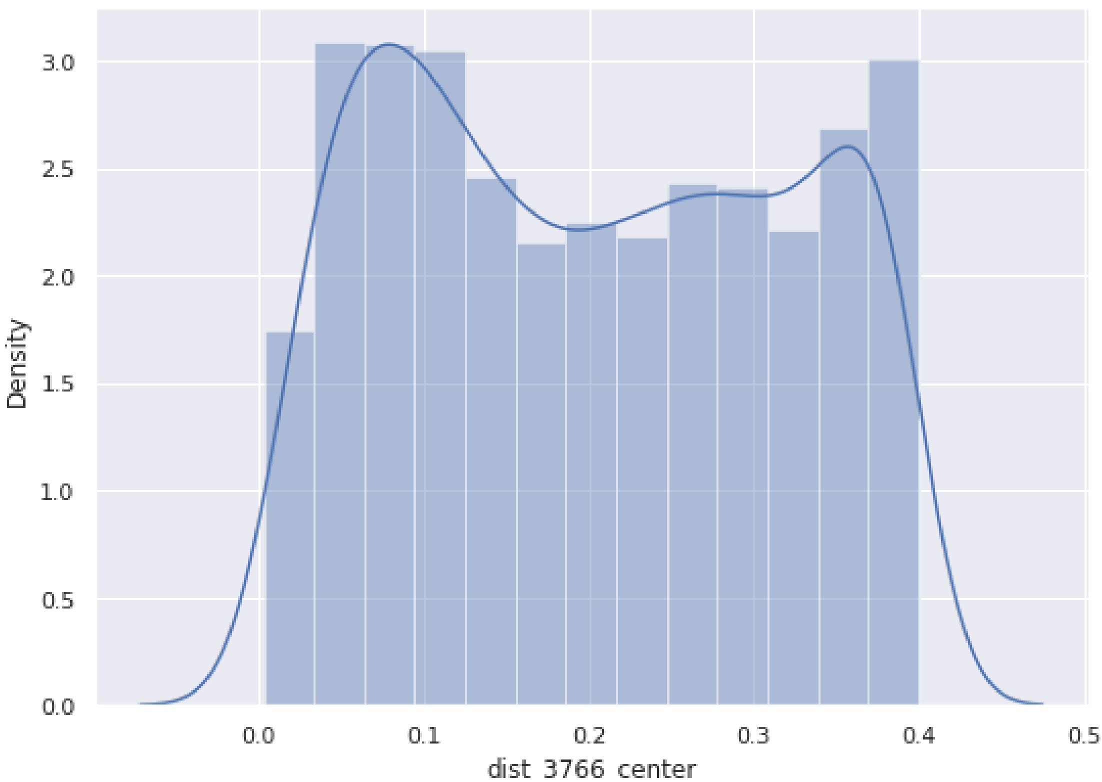
```
Out[498]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4e6e63c048>
```



```
In [499]: # pd_prof.ProfileReport(potentialMember)

In [500]: #files.download('NGC_3766_cantat.csv')

          #files.download('NGC_3766_membership_prob.csv')

In [501]: sns.distplot(concatenated['dist_3766_center'])
```

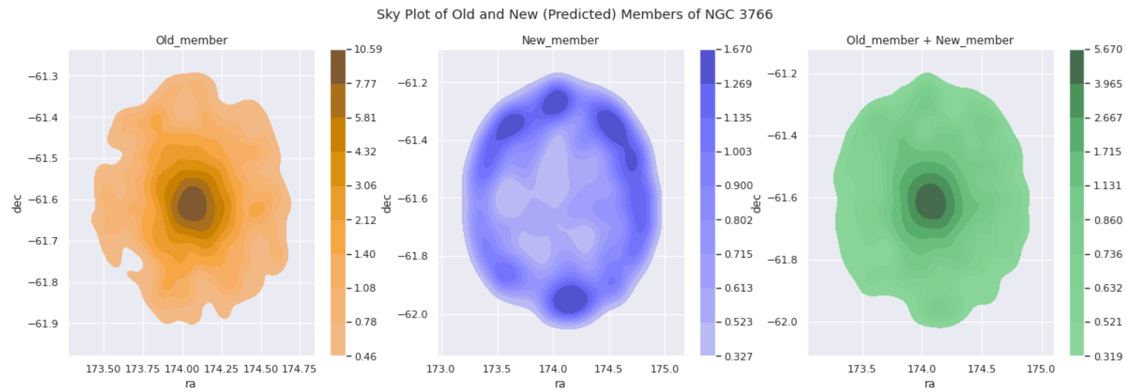
```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
```

```
Out[501]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4e65d0dd30>
```



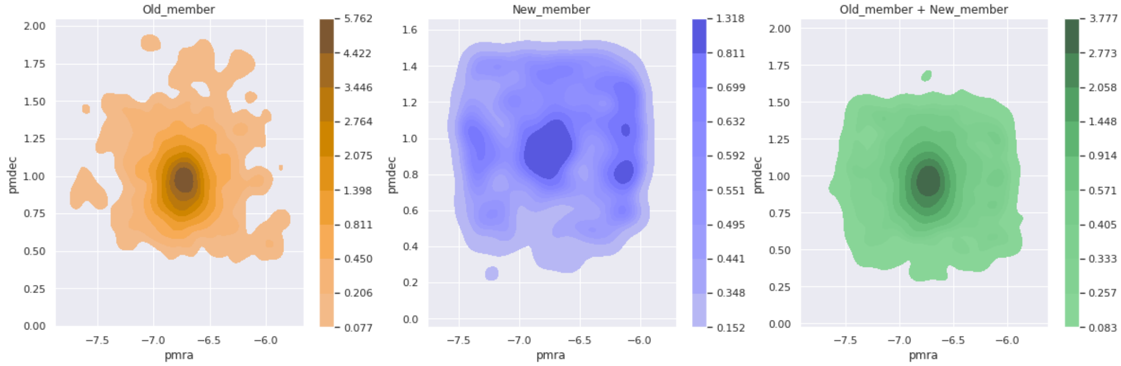

```
In [528]: fig, axes = plt.subplots(1, 3, figsize=(20,6))
          fig.suptitle('Sky Plot of Old and New (Predicted) Members of NGC 3766')

          sns.kdeplot(x='ra', y='dec', data = member, shade=True, color = 'orange',
                      bw_method = 0.20, cbar=True, ax=axes[0])
          axes[0].set_title('Old_member')
```

```
sns.kdeplot(x='ra', y='dec', data = potentialMember, shade=True, color = 'blue',
            bw_method = 0.20, cbar=True, ax=axes[1])
axes[1].set_title('New_member')

sns.kdeplot(x='ra', y='dec', data = concatenated, shade=True, color = 'g',
            bw_method = 0.20, cbar=True, ax=axes[2])
axes[2].set_title('Old_member + New_member')

plt.show()
```




```
In [527]: fig, axes = plt.subplots(1, 3, figsize=(20,6))
          fig.suptitle('Proper Motion Plot of Old and New (Predicted) Members of NGC 3766')

          sns.kdeplot(x='pmra', y='pmdec', data = member, shade=True, color = 'orange',
                      bw_method = 0.20, cbar=True, ax=axes[0])
          axes[0].set_title('Old_member')

          sns.kdeplot(x='pmra', y='pmdec', data = potentialMember, shade=True, color = 'blue',
                      bw_method = 0.20, cbar=True, ax=axes[1])
          axes[1].set_title('New_member')

          sns.kdeplot(x='pmra', y='pmdec', data = concatenated, shade=True, color = 'g',
                      bw_method = 0.20, cbar=True, ax=axes[2])
          axes[2].set_title('Old_member + New_member')

          plt.show()
```

Proper Motion Plot of Old and New (Predicted) Members of NGC 3766



In [510]: