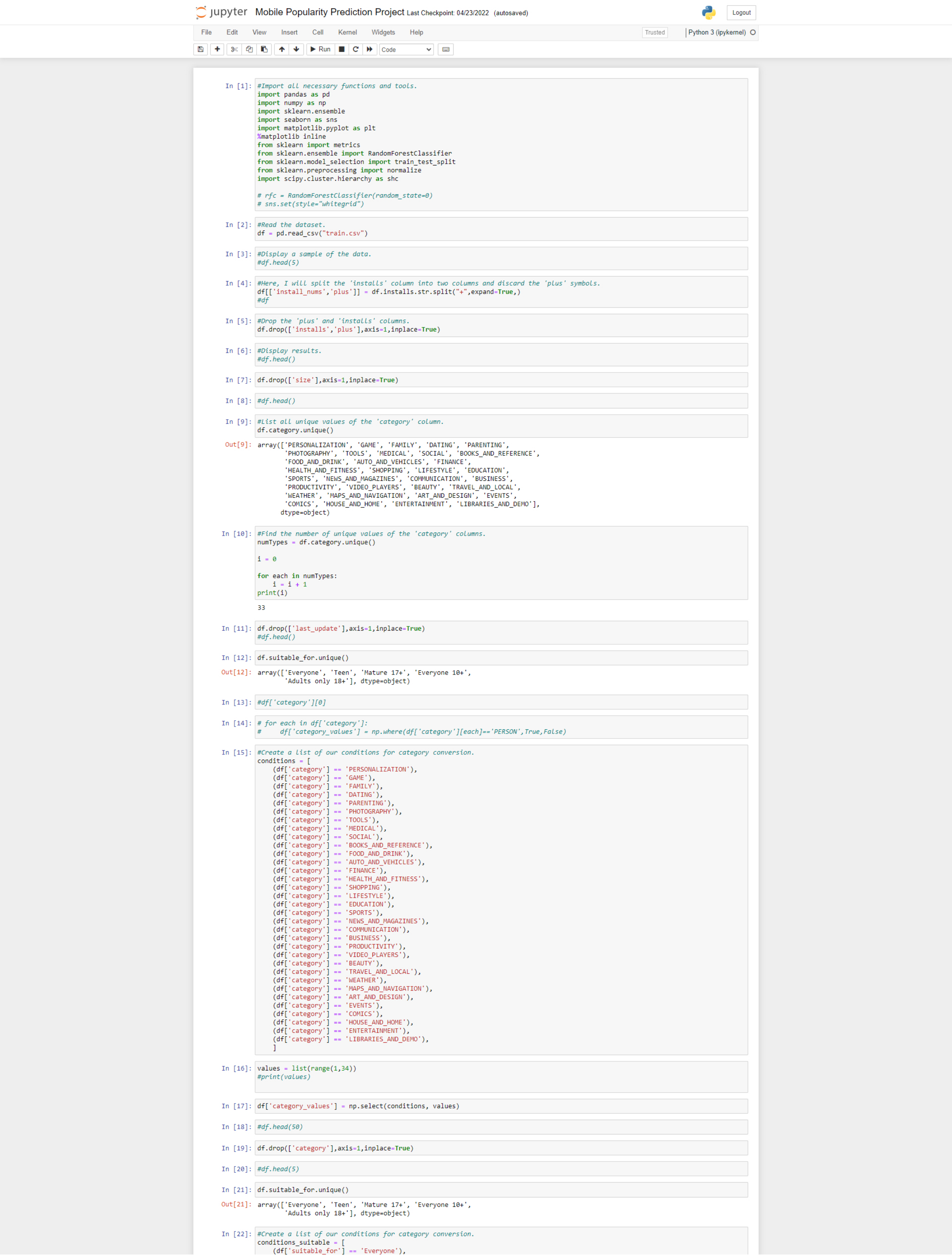


Mobile Popularity Prediction Project Last Checkpoint: 04/23/2022 (autosaved)

In [1]:
```python
#Import all necessary functions and tools.
import pandas as pd
import numpy as np
import sklearn.ensemble
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import metrics
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import normalize
import scipy.cluster.hierarchy as shc

# rfc = RandomForestClassifier(random_state=0)
# sns.set(style="whitegrid")
```

In [2]:
```python
#Read the dataset.
df = pd.read_csv("train.csv")
```

In [3]:
```python
#Display a sample of the data.
#df.head(5)
```

In [4]:
```python
#Here, I will split the 'installs' column into two columns and discard the 'plus' symbols.
df[['install_nums','plus']] = df.installs.str.split("+",expand=True,)
#df
```

In [5]:
```python
#Drop the 'plus' and 'installs' columns.
df.drop(['installs','plus'],axis=1,inplace=True)
```

In [6]:
```python
#Display results.
#df.head()
```

In [7]:
```python
df.drop(['size'],axis=1,inplace=True)
```

In [8]:
```python
#df.head()
```

In [9]:
```python
#List all unique values of the 'category' column.
df.category.unique()
```

Out[9]:
```
array(['PERSONALIZATION', 'GAME', 'FAMILY', 'DATING', 'PARENTING',
       'PHOTOGRAPHY', 'TOOLS', 'MEDICAL', 'SOCIAL', 'BOOKS_AND_REFERENCE',
       'FOOD_AND_DRINK', 'AUTO_AND_VEHICLES', 'FINANCE',
       'HEALTH_AND_FITNESS', 'SHOPPING', 'LIFESTYLE', 'EDUCATION',
       'SPORTS', 'NEWS_AND_MAGAZINES', 'COMMUNICATION', 'BUSINESS',
       'PRODUCTIVITY', 'VIDEO_PLAYERS', 'BEAUTY', 'TRAVEL_AND_LOCAL',
       'WEATHER', 'MAPS_AND_NAVIGATION', 'ART_AND_DESIGN', 'EVENTS',
       'COMICS', 'HOUSE_AND_HOME', 'ENTERTAINMENT', 'LIBRARIES_AND_DEMO'],
      dtype=object)
```

In [10]:
```python
#Find the number of unique values of the 'category' columns.
numTypes = df.category.unique()

i = 0

for each in numTypes:
    i = i + 1
print(i)
```

```
33
```

In [11]:
```python
df.drop(['last_update'],axis=1,inplace=True)
#df.head()
```

In [12]:
```python
df.suitable_for.unique()
```

Out[12]:
```
array(['Everyone', 'Teen', 'Mature 17+', 'Everyone 10+',
       'Adults only 18+'], dtype=object)
```

In [13]:
```python
#df['category'][0]
```

In [14]:
```python
# for each in df['category']:
#     df['category_values'] = np.where(df['category'][each]=='PERSON',True,False)
```

In [15]:
```python
#Create a list of our conditions for category conversion.
conditions = [
    (df['category'] == 'PERSONALIZATION'),
    (df['category'] == 'GAME'),
    (df['category'] == 'FAMILY'),
    (df['category'] == 'DATING'),
    (df['category'] == 'PARENTING'),
    (df['category'] == 'PHOTOGRAPHY'),
    (df['category'] == 'TOOLS'),
    (df['category'] == 'MEDICAL'),
    (df['category'] == 'SOCIAL'),
    (df['category'] == 'BOOKS_AND_REFERENCE'),
    (df['category'] == 'FOOD_AND_DRINK'),
    (df['category'] == 'AUTO_AND_VEHICLES'),
    (df['category'] == 'FINANCE'),
    (df['category'] == 'HEALTH_AND_FITNESS'),
    (df['category'] == 'SHOPPING'),
    (df['category'] == 'LIFESTYLE'),
    (df['category'] == 'EDUCATION'),
    (df['category'] == 'SPORTS'),
    (df['category'] == 'NEWS_AND_MAGAZINES'),
    (df['category'] == 'COMMUNICATION'),
    (df['category'] == 'BUSINESS'),
    (df['category'] == 'PRODUCTIVITY'),
    (df['category'] == 'VIDEO_PLAYERS'),
    (df['category'] == 'BEAUTY'),
    (df['category'] == 'TRAVEL_AND_LOCAL'),
    (df['category'] == 'WEATHER'),
    (df['category'] == 'MAPS_AND_NAVIGATION'),
    (df['category'] == 'ART_AND_DESIGN'),
    (df['category'] == 'EVENTS'),
    (df['category'] == 'COMICS'),
    (df['category'] == 'HOUSE_AND_HOME'),
    (df['category'] == 'ENTERTAINMENT'),
    (df['category'] == 'LIBRARIES_AND_DEMO'),
    ]
```

In [16]:
```python
values = list(range(1,34))
#print(values)
```

In [17]:
```python
df['category_values'] = np.select(conditions, values)
```

In [18]:
```python
#df.head(50)
```

In [19]:
```python
df.drop(['category'],axis=1,inplace=True)
```

In [20]:
```python
#df.head(5)
```

In [21]:
```python
df.suitable_for.unique()
```

Out[21]:
```
array(['Everyone', 'Teen', 'Mature 17+', 'Everyone 10+',
       'Adults only 18+'], dtype=object)
```

In [22]:
```python
#Create a list of our conditions for category conversion.
conditions_suitable = [
    (df['suitable_for'] == 'Everyone'),
    (df['suitable_for'] == 'Teen'),
    (df['suitable_for'] == 'Mature 17+'),
    (df['suitable_for'] == 'Everyone 10+'),
    (df['suitable_for'] == 'Adults only 18+')
    ]
```

```
In [23]: values_suitable = list(range(1,6))
```

```
In [24]: df['suitable_values'] = np.select(conditions_suitable, values_suitable)
```

```
In [25]: #df.head()
```

```
In [26]: #Here, I will split the 'price' column into two columns and discard the 'dollar' symbols.
         df[['dollar','price_real']] = df.price.str.split("$",expand=True,)
         #df.head()
```

```
In [27]: df.drop(['price','dollar'],axis=1,inplace=True)
```

```
In [28]: #df.head(5)
```

```
In [29]: df.drop(['suitable_for'],axis=1,inplace=True)
```

```
In [30]: #df.head(5)
```

```
In [31]: df['prices'] = np.where(df['price_real'] == 'None', '0',df['price_real'])
         #df.head()
```

```
In [32]: df.drop(['price_real','prices'],axis=1,inplace=True)
```

```
In [33]: #df.head()
```

```
In [34]: df.drop(['latest_ver'],axis=1,inplace=True)
```

```
In [35]: #df.head()
```

```
In [36]: #Create a list of our conditions for category conversion.
         conditions_pop = [
             (df['popularity'] == 'High'),
             (df['popularity'] == 'Low')
         ]
```

```
In [37]: values_popularity = list(range(0,2))
```

```
In [38]: df['popularity_val'] = np.select(conditions_pop, values_popularity)
```

```
In [39]: #df.head(50)
```

```
In [40]: df.drop(['popularity'],axis=1,inplace=True)
```

```
In [41]: #df.head()
```

```
In [ ]:
```

```
In [42]: df.drop(['install_nums'],axis=1,inplace=True)
         # = df['install_nums'].astype(int)
```

```
In [43]: x=df.drop(['app_id','popularity_val'],axis=1)
         y=df.popularity_val
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.35, random_state=12340)
         model = RandomForestClassifier(n_estimators=20)
         model.fit(x_train, y_train)
         model.score(x_test, y_test)
         y_pred = model.predict(x_test)
         cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
         print(cnf_matrix)
```

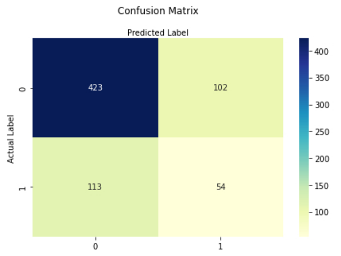
```
[[423 102]
 [113  54]]
```

```
In [44]: print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
         print("Precision:",metrics.precision_score(y_test, y_pred))
         print("Recall:",metrics.recall_score(y_test, y_pred))
```

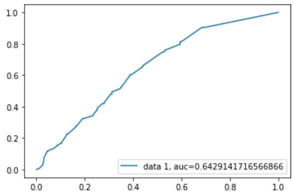
```
Accuracy: 0.6893063583815029
Precision: 0.34615384615384615
Recall: 0.32335329341317365
```

```
In [45]: class_names=[0,1] # name  of classes
         fig, ax = plt.subplots()
         tick_marks = np.arange(len(class_names))
         plt.xticks(tick_marks, class_names)
         plt.yticks(tick_marks, class_names)
         # create heatmap
         sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
         ax.xaxis.set_label_position("top")
         plt.tight_layout()
         plt.title('Confusion Matrix', y=1.1)
         plt.ylabel('Actual Label')
         plt.xlabel('Predicted Label')
```

```
Out[45]: Text(0.5, 257.44, 'Predicted Label')
```



```
In [46]: y_pred_proba = model.predict_proba(x_test)[::,1]
         fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
         auc = metrics.roc_auc_score(y_test, y_pred_proba)
         plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
         plt.legend(loc=4)
         plt.show()
```



```
In [ ]:
```