```
import matplotlib.pyplot as plt
import matplotlib
import numpy as np
import pandas as pd
from IPython.display import display
```

```
data = pd.read_csv('/content/LosVegasDataset.csv')
data.head()
```

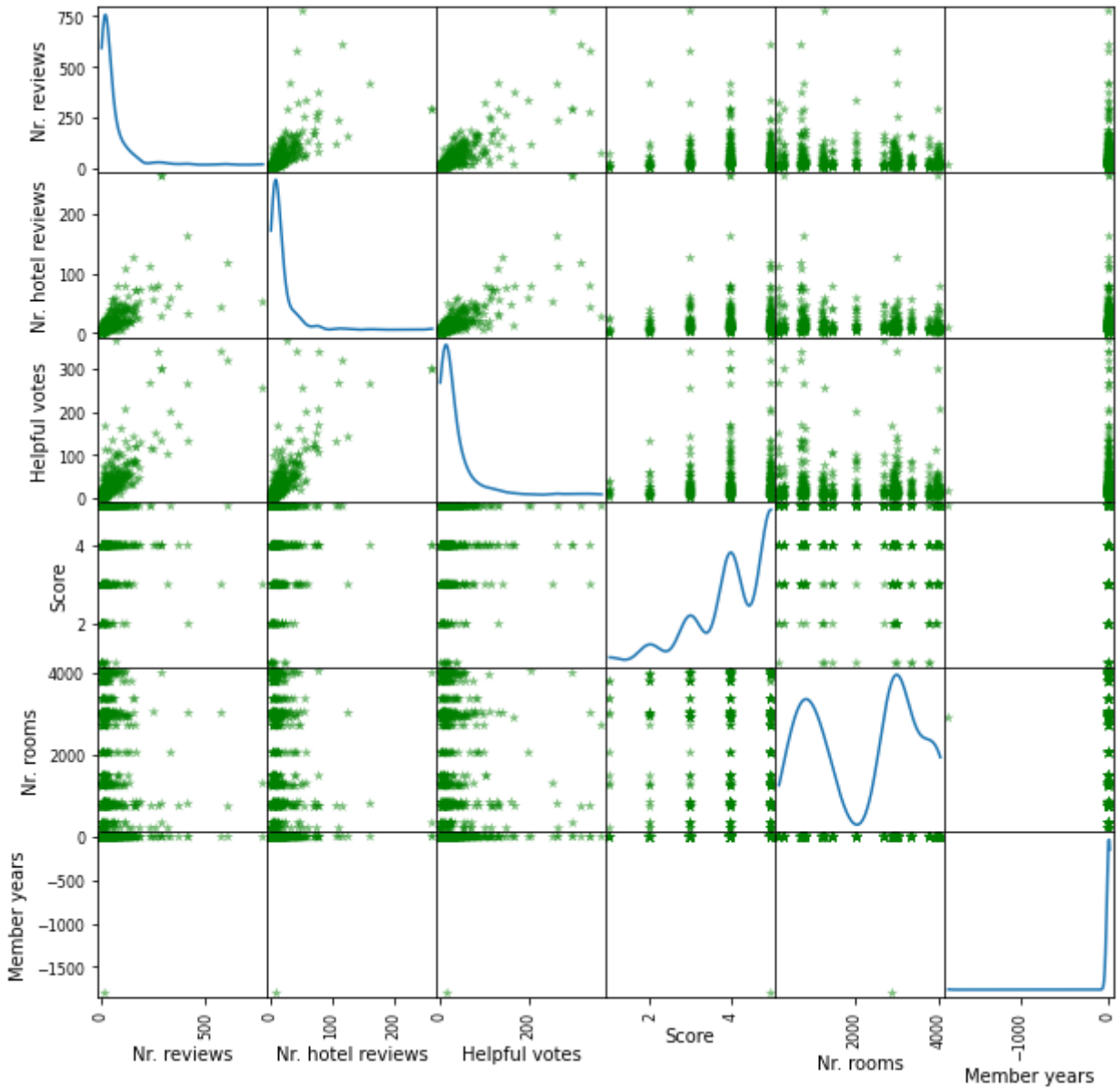
| | User country | Nr. reviews | Nr. hotel reviews | Helpful votes | Score | Period of stay | Traveler type | Pool | Gym | Tennis court | Spa |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | USA | 11 | 4 | 13 | 5 | Dec-Feb | Friends | NO | YES | NO | NO |
| **1** | USA | 119 | 21 | 75 | 3 | Dec-Feb | Business | NO | YES | NO | NO |

```
display(data.describe(include=[np.number]))
display(data.describe(exclude=[np.number]))
```

Nr. Nr. hotel Helpful Score Nr. rooms Member

```
##Scatter matrix of features

pd.plotting.scatter_matrix(data,
                           figsize=(10,10),
                           diagonal='kde',
                           s=40,
                           alpha=0.5,
                           marker='*',
                           color='green');
```




> For the above dataset y-label will be the `score` and rest of the features will be used score prediction

```
## list of categorical variables which need encoding
categorical = list(data.select_dtypes(include=['object']).columns.values)
categorical
```

```
['User country',
 'Period of stay',
 'Traveler type',
 'Pool',
 'Gym',
 'Tennis court',
 'Spa',
 'Casino',
 'Free internet',
 'Hotel name',
 'Hotel stars',
 'User continent',
 'Review month',
 'Review weekday']
```

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()

# seasons in place of months

['Dec-Feb' 'Mar-May' 'Jun-Aug' 'Sep-Nov']
data['Period of stay'] = data['Period of stay'].map({'Dec-Feb':'winter', 'Mar-May':'spring',

for i in range(0, len(categorical)):
  #  data[categorical[i]] = le.fit_transform(data[categorical[i]])
    print(data[categorical[i]].unique())
```

```
['USA' 'UK' 'Canada' 'India' 'Australia' 'New Zeland' 'Ireland' 'Egypt'
 'Finland' 'Kenya' 'Jordan' 'Netherlands' 'Syria' 'Scotland'
 'South Africa' 'Swiss' 'United Arab Emirates' 'Hungary' 'China' 'Greece'
 'Mexico' 'Croatia' 'Germany' 'Malaysia' 'Thailand' 'Phillippines'
 'Israel' 'India ' 'Belgium' 'Puerto Rico' 'Switzerland' 'Norway' 'France'
 'Spain' 'Singapore' 'Brazil' 'Costa Rica' 'Iran' 'Saudi Arabia'
 'Honduras' 'Denmark' 'Taiwan' 'Hawaii' 'Kuwait' 'Czech Republic' 'Japan'
 'Korea' 'Italy']
['winter' 'spring' 'summer' 'autumn']
['Friends' 'Business' 'Families' 'Solo' 'Couples']
['NO' 'YES']
['YES' 'NO']
['NO' 'YES']
['NO' 'YES']
['YES' 'NO']
['YES' 'NO']
['Circus Circus Hotel & Casino Las Vegas' 'Excalibur Hotel & Casino'
 'Monte Carlo Resort&Casino' 'Treasure Island- TI Hotel & Casino'
 'Tropicana Las Vegas - A Double Tree by Hilton Hotel' 'Caesars Palace'
 'The Cosmopolitan Las Vegas' 'The Palazzo Resort Hotel Casino'
 'Wynn Las Vegas' 'Trump International Hotel Las Vegas' 'The Cromwell'
 'Encore at wynn Las Vegas' 'Hilton Grand Vacations on the Boulevard'
 "Marriott's Grand Chateau" 'Tuscany Las Vegas Suites & Casino'
 'Hilton Grand Vacations at the Flamingo' 'Wyndham Grand Desert'
 'The Venetian Las Vegas Hotel' 'Bellagio Las Vegas' 'Paris Las Vegas'
 'The Westin las Vegas Hotel Casino & Spa']
['3' '4' '5' '4,5' '3,5']
['North America' 'Europe' 'Asia' 'Oceania' 'Africa' 'South America']
```

```
['January' 'February' 'March' 'April' 'May' 'June' 'July' 'August'
 'September' 'October' 'November' 'December']
['Thursday' 'Friday' 'Saturday' 'Tuesday' 'Wednesday' 'Sunday' 'Monday']
```

```
#Encoding categorical features with numbers

for i in range(0, len(categorical)):
    data[categorical[i]] = le.fit_transform(data[categorical[i]])

data.head()
```

| | User country | Nr. reviews | Nr. hotel reviews | Helpful votes | Score | Period of stay | Traveler type | Pool | Gym | Tennis court | Spa |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 46 | 11 | 4 | 13 | 5 | 3 | 3 | 0 | 1 | 0 | 0 |
| **1** | 46 | 119 | 21 | 75 | 3 | 3 | 0 | 0 | 1 | 0 | 0 |
| **2** | 46 | 36 | 9 | 25 | 5 | 1 | 2 | 0 | 1 | 0 | 0 |
| **3** | 45 | 14 | 7 | 14 | 4 | 1 | 3 | 0 | 1 | 0 | 0 |
| **4** | 3 | 5 | 5 | 2 | 4 | 1 | 4 | 0 | 1 | 0 | 0 |

**Prepare train and test sets**

```
## prepare train and test labels

from sklearn.model_selection import train_test_split
X= data.drop(['Score'], axis=1) ## remove score label from data
y = data['Score']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

**Recursive feature elimination (RFE) to select features by recursively considering smaller and smaller sets of features**

```
##Applying random forest classifier on features to class

## Using feature selection pipeline and classification

from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import SelectFromModel
from sklearn.feature_selection import RFE
from sklearn.linear_model import Ridge


rfe = RFE(estimator = Ridge(), n_features_to_select = 12)
rfe.fit(X_train, y_train)
```

```
feature_list = pd.DataFrame({'col':list(X_train.columns.values),'sel':list(rfe.support_ *1)})
print("*Most contributing features in Score*")
cols=feature_list[feature_list.sel==1].col.values
print(feature_list[feature_list.sel==1].col.values)
cols=feature_list[feature_list.sel==1].col.values
```

```
*Most contributing features in Score*
['Period of stay' 'Traveler type' 'Pool' 'Gym' 'Spa' 'Casino'
 'Free internet' 'Hotel stars' 'User continent' 'Member years'
 'Review month' 'Review weekday']
```

```
## Subset train data based on selected features
X_sel = pd.DataFrame(X_train, columns=(cols))
X_sel_t = pd.DataFrame(X_test, columns=(cols[:13]))
```

**Using selected features to fit the model for Score prediction**

> Using random forest classifier and kNN to predict score

**Using Random forest**

```
clf = RandomForestClassifier(max_depth=5, random_state=0)
clf.fit(X_sel, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=5, 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=False, random_state=0, verbose=0,
                       warm_start=False)
```

**Using Support Vector Classifier**

```
from sklearn.svm import SVC
clf2=SVC()
clf2.fit(X_sel,y_train)
```

```
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=None, shrinking=True,
    tol=0.001, verbose=False)
```

**Using KNN for score prediction**

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=10)
```

```
knn.fit(X_sel, y_train)
print(knn.score(X_sel_t, y_test))
print(clf.score(X_sel_t, y_test))
print(clf2.score(X_sel_t,y_test))
```

```
0.42105263157894735
0.5
0.45394736842105265
```

**Random Forest gave better score than KNN on test data**

- Using Random forest for `Score` Prediction

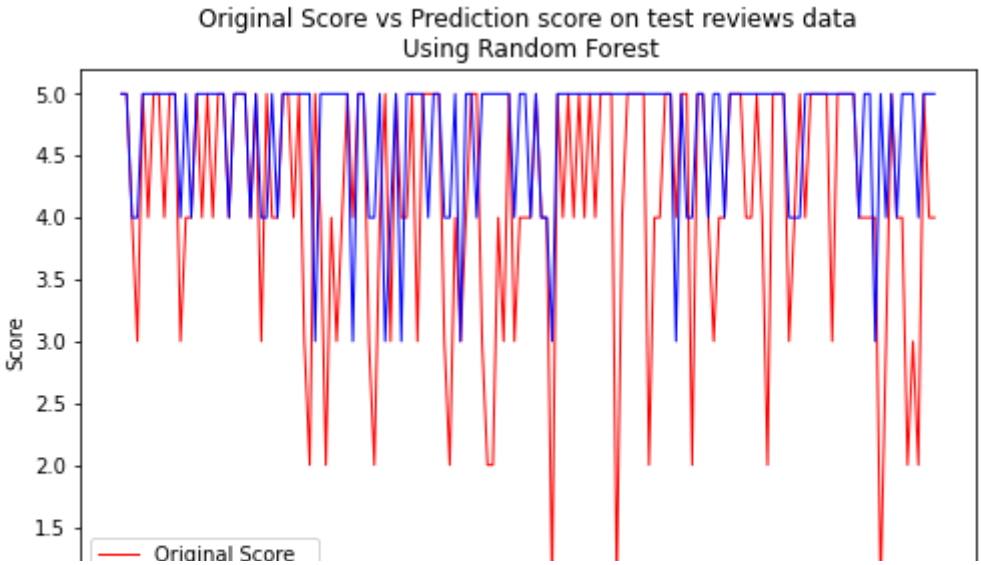
```
p = (list(clf.predict(X_sel_t)))
len(p)
```

```
152
```

```
Predictions = X_sel_t
Predictions['Original_Score']= y_test
Predictions['pred_score'] = p
Predictions.head()
```

| | Period of stay | Traveler type | Pool | Gym | Spa | Casino | Free internet | Hotel stars | User continent | Member years | Revi mon |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **173** | 1 | 2 | 1 | 1 | 1 | 1 | 1 | 4 | 3 | 8 | |
| **274** | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 4 | 2 | 2 | |
| **489** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 3 | 3 | |
| **72** | 3 | 0 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 6 | |

```
plt.figure(figsize=(8, 5))
ax = plt.subplot()
d = list(range(0,len(Predictions)))
p1 = plt.plot(d,Predictions['Original_Score'],'r-', label="Original Score", linewidth= 1 )#,data
p2 = plt.plot(d,Predictions['pred_score'],'b-', label="Predicted Score", linewidth= 1)
ax.set_title('Original Score vs Prediction score on test reviews data\n Using Random Forest
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, labels)
ax.set_xlabel('review Id')
ax.set_ylabel('Score')
plt.show()
```




**Summary**

The best prediction score achived using Random Forest with 12 features out of 19, is **46%** which is fairly low as the data is not so rich in my opinion

However Random Forest performed better than KNN in score prediction

✓ 0s completed at 11:19 PM