yamsafer

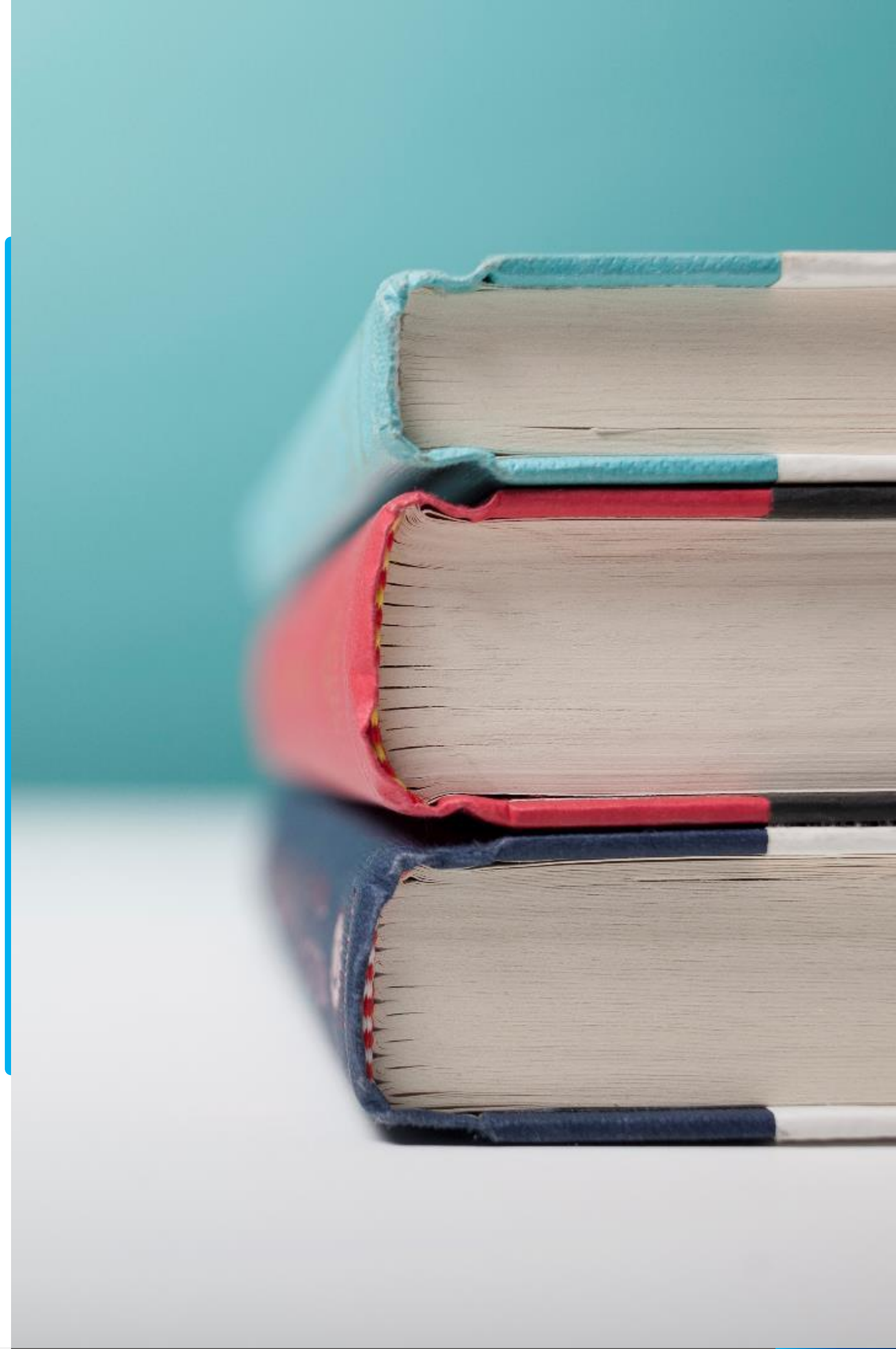
Design and code bootcamp 2018

Rami Salman & Tariq Hamayel

Introduction

what is the objective?

- The objective is to analyse a set of data then create a model to predict another set cancellation based on the other features



Exploratory of data :

hotel.stars: Descending	number_of_rooms: Descending	nights: Descending	checkin_date: Descending	is_prepaid: Descending	is_cardless: Descending	includes_weekend: Descending	hotel.id: Descending	
0	1	1	July 9, 2017	0	1	0	822620	**0**
4	1	1	December 4, 2017	0	0	0	27032	**1**
5	1	7	June 28, 2017	0	0	1	452806	**2**
3	1	3	February 26, 2017	0	0	0	621754	**3**
2	1	1	October 15, 2017	1	0	0	205042	**4**

hotel.city_en.keyword: Descending	hotel.country_en.keyword: Descending	customer.country_code.keyword: Descending	hotel.type.keyword: Descending
Jeddah	Saudi Arabia	SA	Hotel
Cairo	Egypt	SA	Hotel
Nusa Dua	Indonesia	SA	Hotel Resort
Dubai	United Arab Emirates	SA	Hotel
Abu Dhabi	United Arab Emirates	AE	Hotel

- Given data example:

cancelled: Descending	created_at per day	customer.platform.keyword: Descending
1	July 3, 2017	iPhoneApp
0	December 4, 2017	iPhoneApp
0	June 19, 2017	Chrome
1	February 20, 2017	iPhoneApp
0	October 9, 2017	AndroidApp

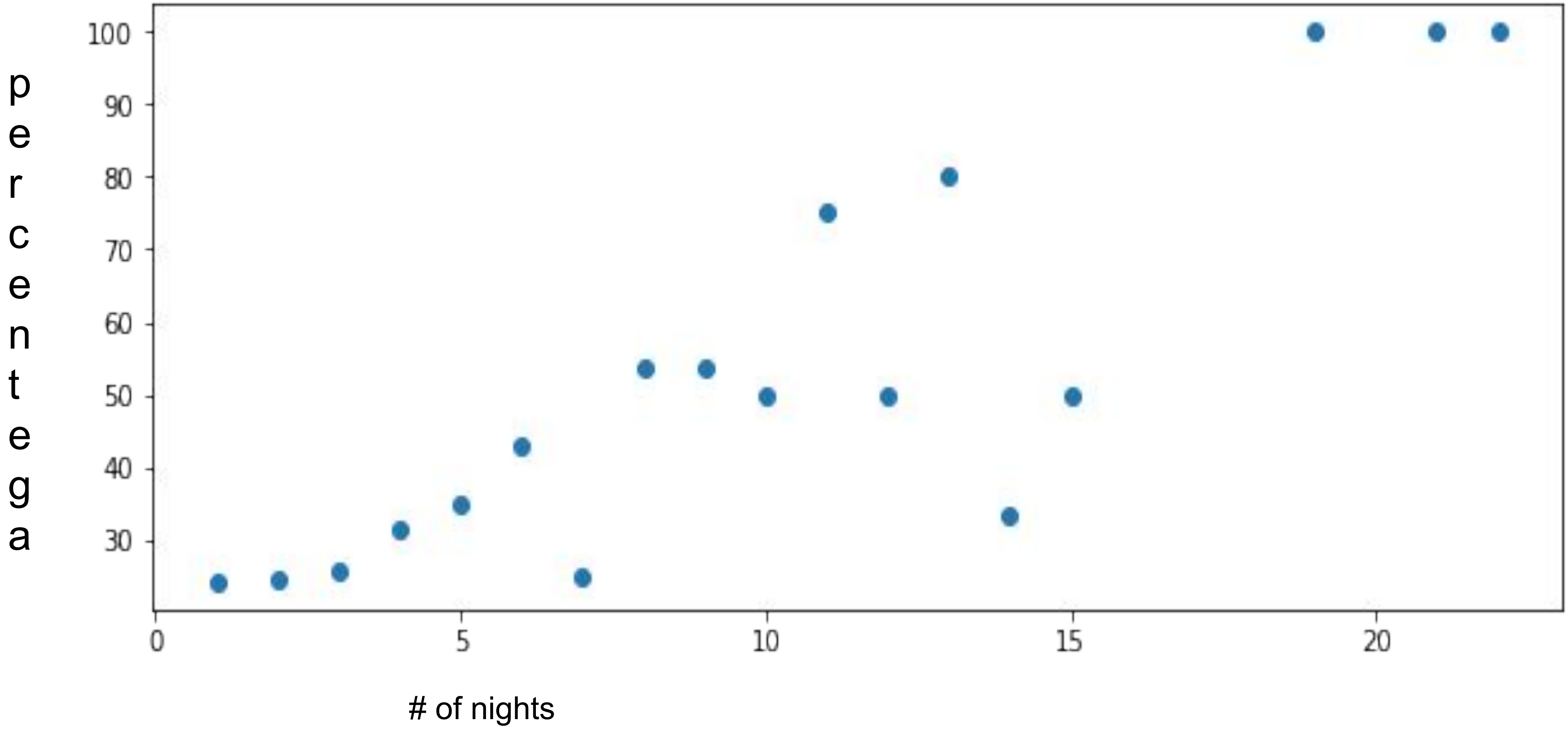
So our task is to create a model based on the features that affect the cancellation

Visualization of data :

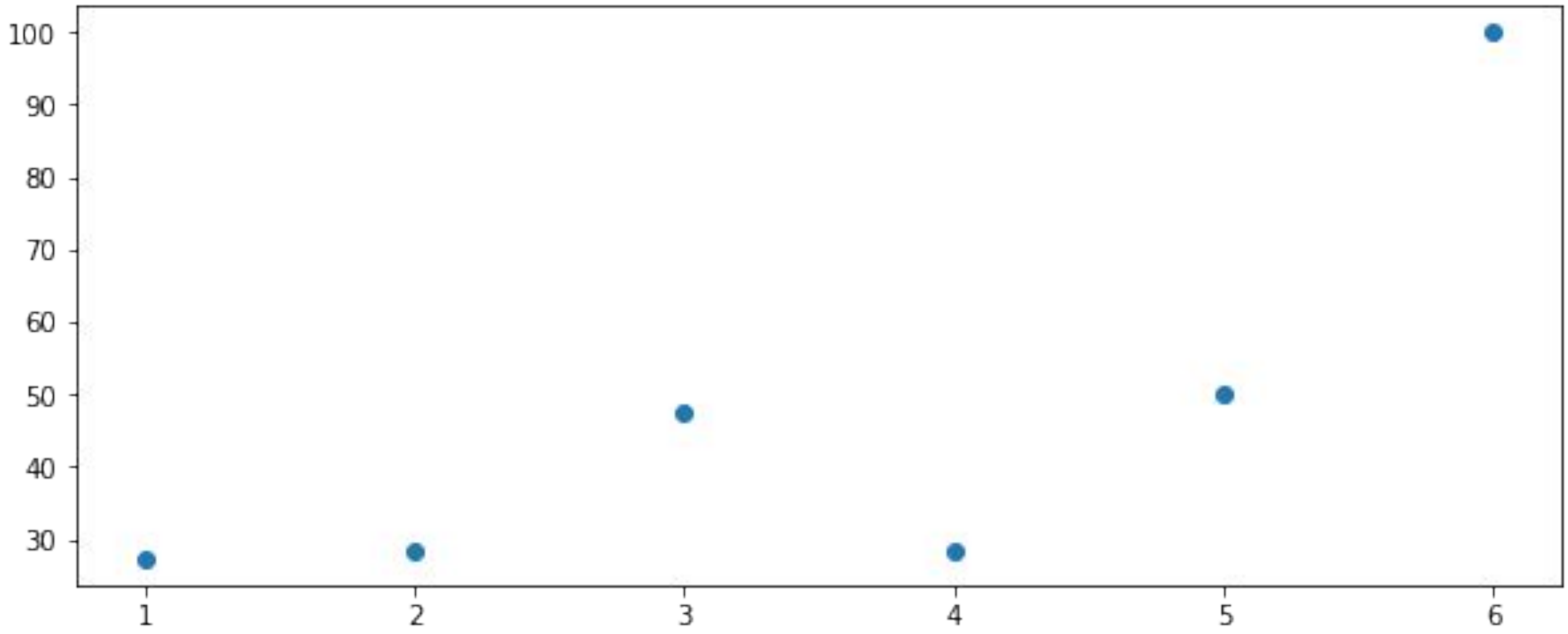
We plotted the relation between labels of cancellation and the features to notice the most features that affect the cancellation , next slides some examples


cancelled
not cancelled
24.92
cancelled
75.08

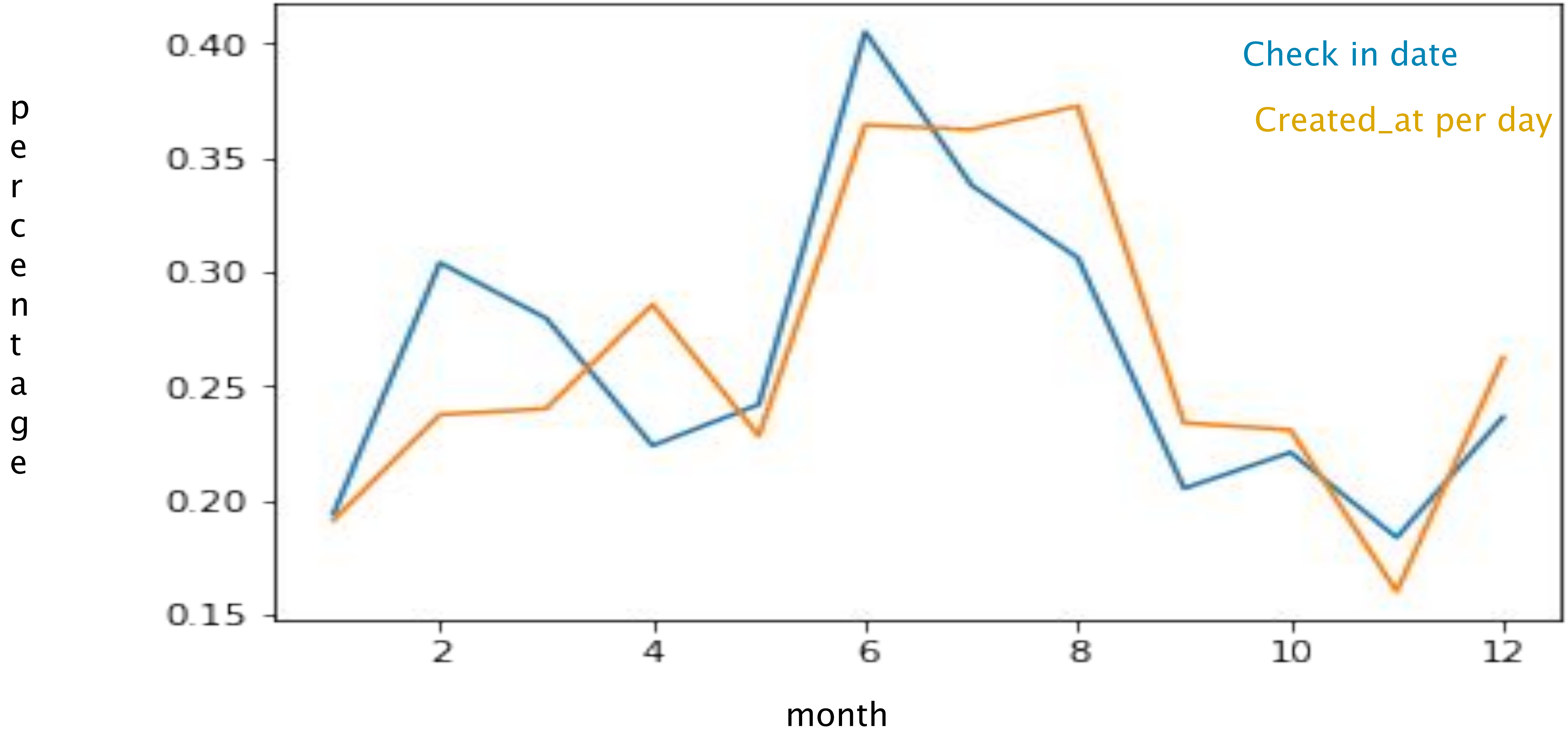
number of nights and cancelation



number of rooms and cancelation

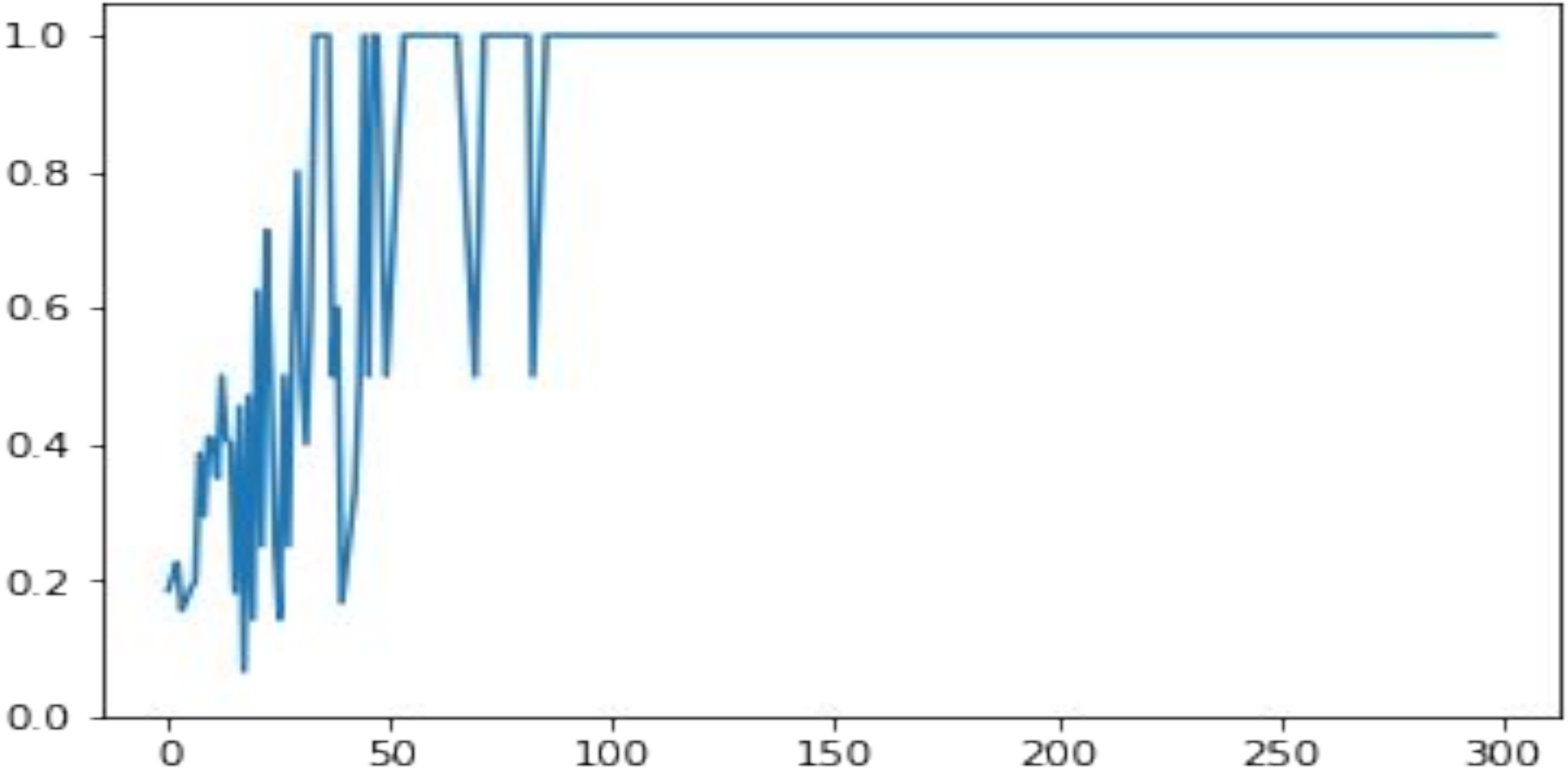


Labels vs. features :



difference of number of days

p
e
r
c
e
n
t
a
g
e



days difference

Normalization For the data

- WE normalized the Different feature depends data type for each feature

```
def encode(data):
    values = array(data)
#     print(values)
    # integer encode
    label_encoder = LabelEncoder()
    integer_encoded = label_encoder.fit_transform(values)
#     print(integer_encoded)
    # binary encode
    onehot_encoder = OneHotEncoder(sparse=False)
    integer_encoded = integer_encoded.reshape(len(data), 1)
    onehot_encoded = onehot_encoder.fit_transform(integer_encoded)
#     print(onehot_encoded.size)
    return integer_encoded, label_encoder

def decode(data,index, label_encoder):
    values = array(data)
    return label_encoder.inverse_transform([argmax(data[index, :])])

encode1,lab=encode(dataSet['hotel.id: Descending'])
df1=pd.DataFrame(encode1)
# encoded+=(encode1)

encode2,lab=encode(dataSet['hotel.type.keyword: Descending'])
df2=pd.DataFrame(encode2)
# encoded+=(encode2)

encode3,lab=encode(dataSet['customer.country_code.keyword: Descending'])
df3=pd.DataFrame(encode3)
# encoded+=(encode3)

encode4,lab=encode(dataSet['hotel.country_en.keyword: Descending'])
df4=pd.DataFrame(encode4)
# encoded+=(encode4)

encode5,lab=encode(dataSet['hotel.city_en.keyword: Descending'])
df5=pd.DataFrame(encode5)
# encoded+=(encode5)

encode6,lab=encode(dataSet['customer.platform.keyword: Descending'])
df6=pd.DataFrame(encode6)
```

```
df3=pd.DataFrame(encode3)
# encoded+=(encode3)

encode4,lab=encode(dataSet['hotel.country_en.keyword: Descending'])
df4=pd.DataFrame(encode4)
# encoded+=(encode4)

encode5,lab=encode(dataSet['hotel.city_en.keyword: Descending'])
df5=pd.DataFrame(encode5)
# encoded+=(encode5)

encode6,lab=encode(dataSet['customer.platform.keyword: Descending'])
df6=pd.DataFrame(encode6)

encArray=np.concatenate((encode1,encode2,
                         encode3,encode4,encode5,encode6),axis=1)

df1=np.concatenate((df1,df2,df3,df4,df5,df6),axis=1)
dataArray=np.concatenate((dataSet[numericHeaders],dataSet[booleanHeaders],
                         created_at_per_daydf,checkin_datedf,df1),axis=1)

labelsArray=dataSet[labelHeaders]
print dataArray.shape
print labelsArray.shape
```

```
(1201L, 14L)
(1201, 1)
```

split

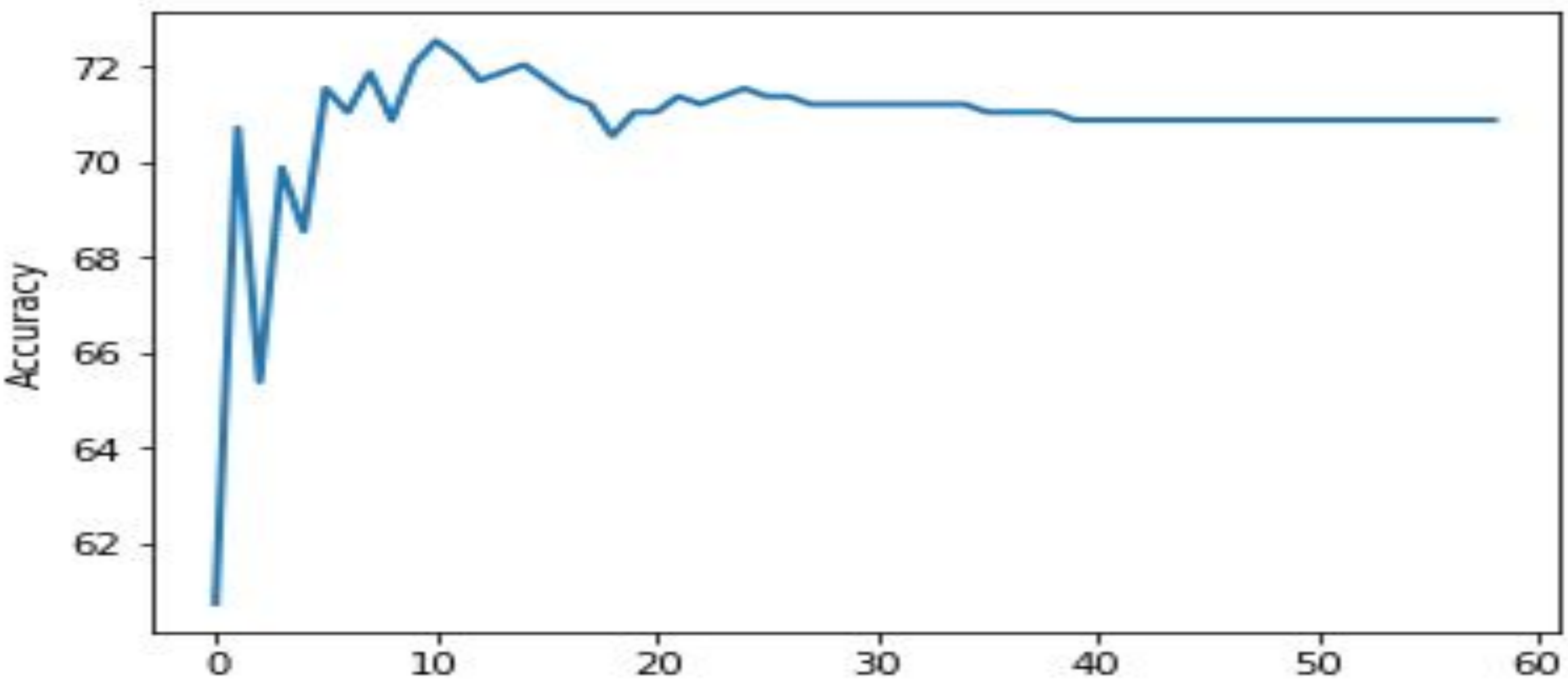
```
[22]: from sklearn.model_selection import train_test_split

xTrain,xValid,yTrain,yValid=train_test_split(dataArray,labelsArray,test_size=0.35,shuffle=True)
print xTrain[0]
print xValid.shape
print yTrain.shape
print yValid.shape
```

```
[5.0000000e+00 3.0000000e+00 1.0000000e+00 0.0000000e+00 0.0000000e+00
 1.0000000e+00 1.5032628e+09 1.5042996e+09 2.0000000e+02 5.0000000e+00
 0.0000000e+00 4.1000000e+01 4.7000000e+01 9.0000000e+00]
(421L, 14L)
(780, 1)
(421, 1)
```

Knn- graph

-hotel starts , is Prepared, number of night



```
model = KNN(n_neighbors = 15 )
x_train=dataSet[['diffrence','number_of_rooms: Descending'
                 ,'nights: Descending']]
y_train=dataSet['cancelled: Descending']
x_test=df2[['diffrence','number_of_rooms: Descending','nights: Descending']]

model.fit(x_train, y_train)
prediction = model.predict(x_test)
xtest = pd.DataFrame({'prediction' : prediction,
                                  },
                  columns=['prediction'])
xtest_export = xtest[['prediction']]
xtest_export.to_csv('prediction.csv', index = True)
print prediction
```

logisitic regression

```
: from sklearn.linear_model import LogisticRegression
  from sklearn.datasets.samples_generator import make_blobs
  from sklearn.linear_model import LogisticRegression
  classifier = LogisticRegression(random_state = 29)
  classifier.fit(xTrain, yTrain)

  # Predicting the Test set results
  y_pred = classifier.predict(xValid)
  xtest = pd.DataFrame({'prediction' : y_pred},columns=['prediction'])
  plt.plot(y_pred)
  xtest_export = xtest[['prediction']]
  xtest_export.to_csv('c29.csv', index = True)
  # print y_pred
```

SSE and Silhouette errors

20]:
```
clustering_data=np.concatenate((xTrain,xValid),axis=0)[:,:2]
```

21]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

kmeans = KMeans(n_clusters=5).fit(clustering_data) # Learning the cluster cente
print('Cluster centers:')
print(kmeans.cluster_centers_)
print('\nData labels (first 30 samples):')
print(kmeans.labels_[:30])
print('\nSum of Squared Error (SSE) of this particular clustering:')
print(kmeans.inertia_)
print('\nSilhouette Score:')
print(silhouette_score(clustering_data,kmeans.labels_))
```

```
Cluster centers:
[[1.62500000e-02 2.00000000e-02]
 [8.73555166e-01 2.28983499e-16]
 [1.00000000e+00 4.00000000e+00]
 [5.69452450e-01 2.53602305e-02]
 [8.47540984e-01 2.80327869e-01]]

Data labels (first 30 samples):
[1 1 3 1 3 3 1 1 0 1 3 3 4 0 1 3 1 1 0 3 3 4 1 1 1 1 0 1 1 0]

Sum of Squared Error (SSE) of this particular clustering:
15.434508129186483

Silhouette Score:
0.6730734599487378
```

svm and confusing metrices

```
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix

svclassifier = SVC(kernel='linear')
svclassifier.fit(xValid, yValid)
y_pred = svclassifier.predict(xValid)
print(confusion_matrix(yValid,y_pred))
print(classification_report(yValid,y_pred))
```

```
[[283   0]
 [137   1]]
             precision    recall  f1-score   support

          0       0.67      1.00      0.81       283
          1       1.00      0.01      0.01       138

avg / total       0.78      0.67      0.55       421
```

cross validation

```
from sklearn import svm
clf = svm.SVC(kernel='linear', C=5).fit(xTrain, yTrain)
print clf.score(xValid, yValid)
```

```
0.672209026128266
```

Result - -

We find that the most accurate prediction module is Knn