```
In [245]: import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt
          import seaborn as sns
```

```
In [180]: data = pd.read_csv('datasets_13720_18513_insurance.csv')

          print(data.shape)

          data.head()
```

(1338, 7)

Out[180]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

```
In [181]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1338 non-null   int64
 1   sex       1338 non-null   object
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64
 4   smoker    1338 non-null   object
 5   region    1338 non-null   object
 6   charges   1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

```
In [182]: data.describe()
```

Out[182]:

	age	bmi	children	charges
count	1338.000000	1338.000000	1338.000000	1338.000000
mean	39.207025	30.663397	1.094918	13270.422265
std	14.049960	6.098187	1.205493	12110.011237
min	18.000000	15.960000	0.000000	1121.873900
25%	27.000000	26.296250	0.000000	4740.287150
50%	39.000000	30.400000	1.000000	9382.033000
75%	51.000000	34.693750	2.000000	16639.912515
max	64.000000	53.130000	5.000000	63770.428010

```
In [183]: data.isnull().sum()
```

```
Out[183]: age         0
          sex         0
          bmi         0
          children    0
          smoker      0
          region      0
          charges     0
          dtype: int64
```

```
In [184]: data['sex'].value_counts()
```

```
Out[184]: male      676
          female    662
          Name: sex, dtype: int64
```

```
In [185]: data['age'].value_counts()
```

```
Out[185]: 18    69
          19    68
          51    29
          45    29
          46    29
          47    29
          48    29
          50    29
          52    29
          20    29
          26    28
          54    28
          53    28
          25    28
          24    28
          49    28
          23    28
          22    28
          21    28
          27    28
          28    28
          31    27
          29    27
          30    27
          41    27
          43    27
          44    27
          40    27
          42    27
          57    26
          34    26
          33    26
          32    26
          56    26
          55    26
          59    25
          58    25
          39    25
          38    25
          35    25
          36    25
          37    25
          63    23
          60    23
          61    23
          62    23
          64    22
          Name: age, dtype: int64
```

```
In [186]: data['children'].value_counts()
```

```
Out[186]: 0    574
          1    324
          2    240
          3    157
          4     25
          5     18
          Name: children, dtype: int64
```

```
In [187]: data['smoker'].value_counts()
```

```
Out[187]: no     1064
          yes     274
          Name: smoker, dtype: int64
```

```
In [188]: data['region'].value_counts()
```

```
Out[188]: southeast    364
          northwest    325
          southwest    325
          northeast    324
          Name: region, dtype: int64
```

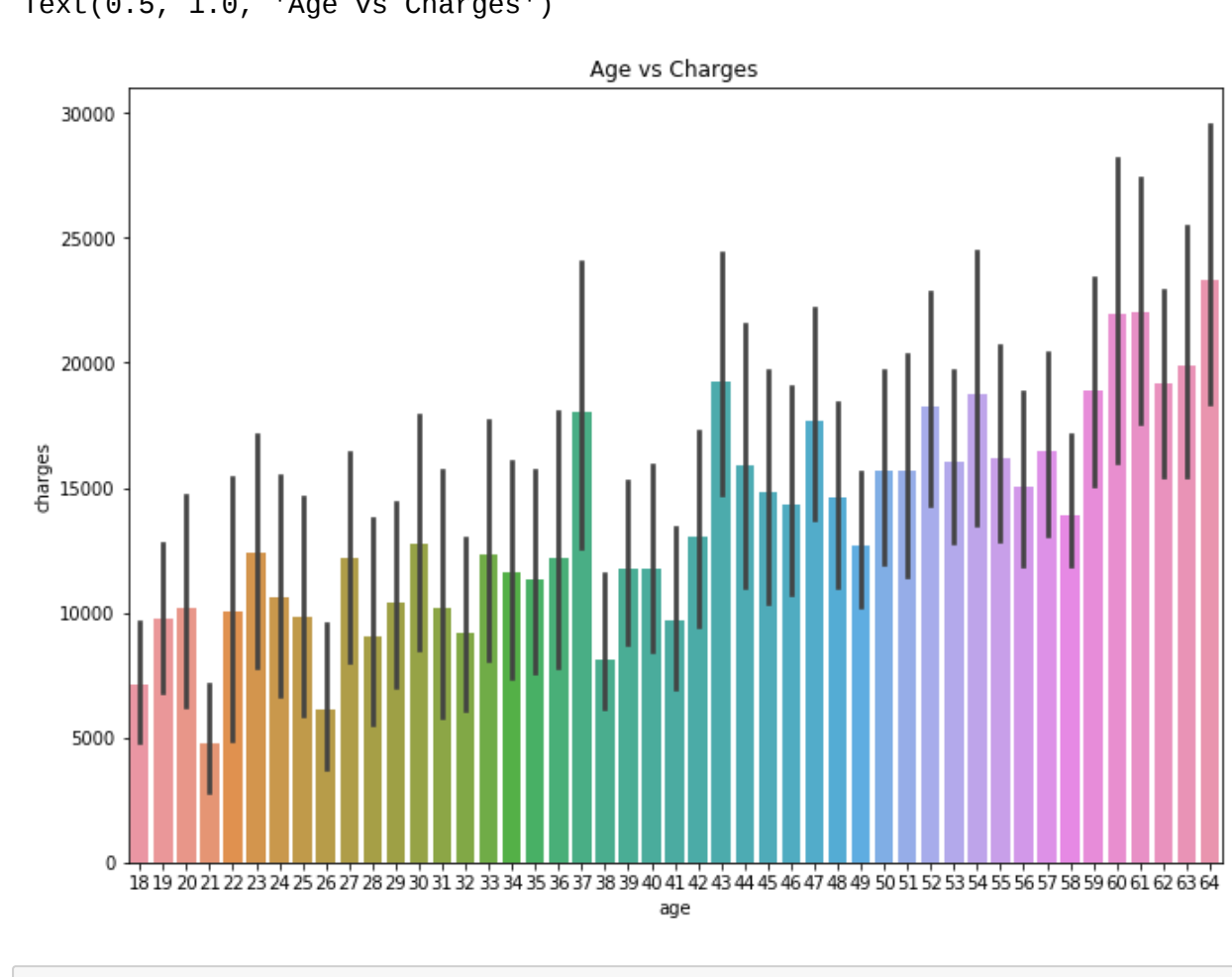
```
In [189]: data.columns
```

```
Out[189]: Index(['age', 'sex', 'bmi', 'children', 'smoker', 'region', 'charges'],
          dtype='object')
```

```
In [190]: plt.figure(figsize = (11, 8))
          sns.barplot(x = 'age', y = 'charges', data = data)

          plt.title("Age vs Charges")
```

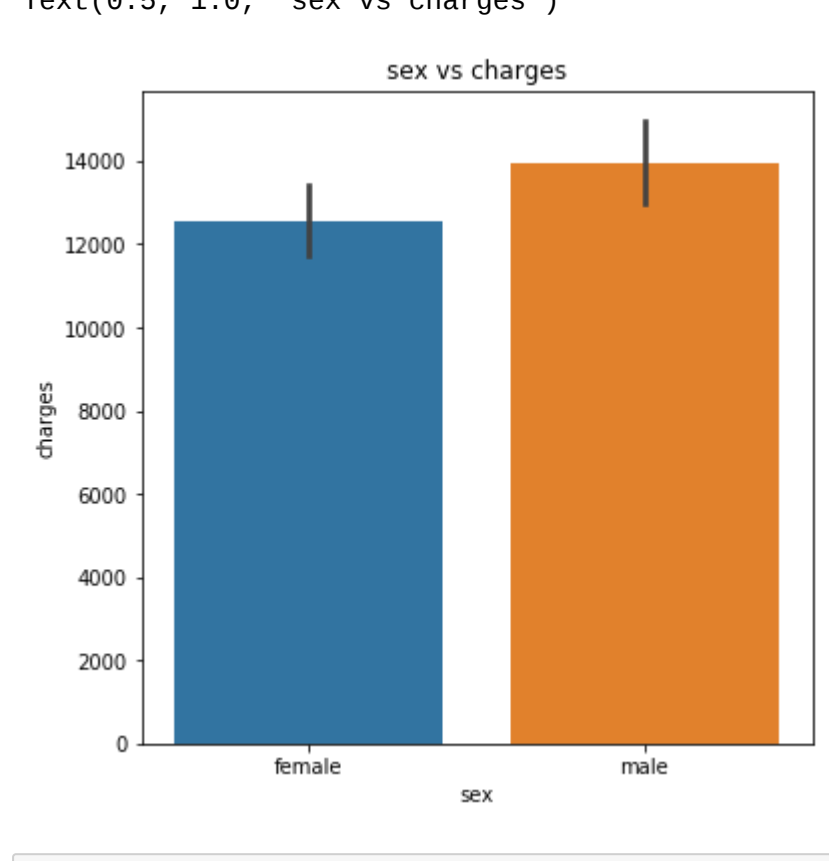
Out[190]: Text(0.5, 1.0, 'Age vs Charges')




```
In [191]: plt.figure(figsize = (6, 6))
          sns.barplot(x = 'sex', y = 'charges', data = data)

          plt.title('sex vs charges')
```

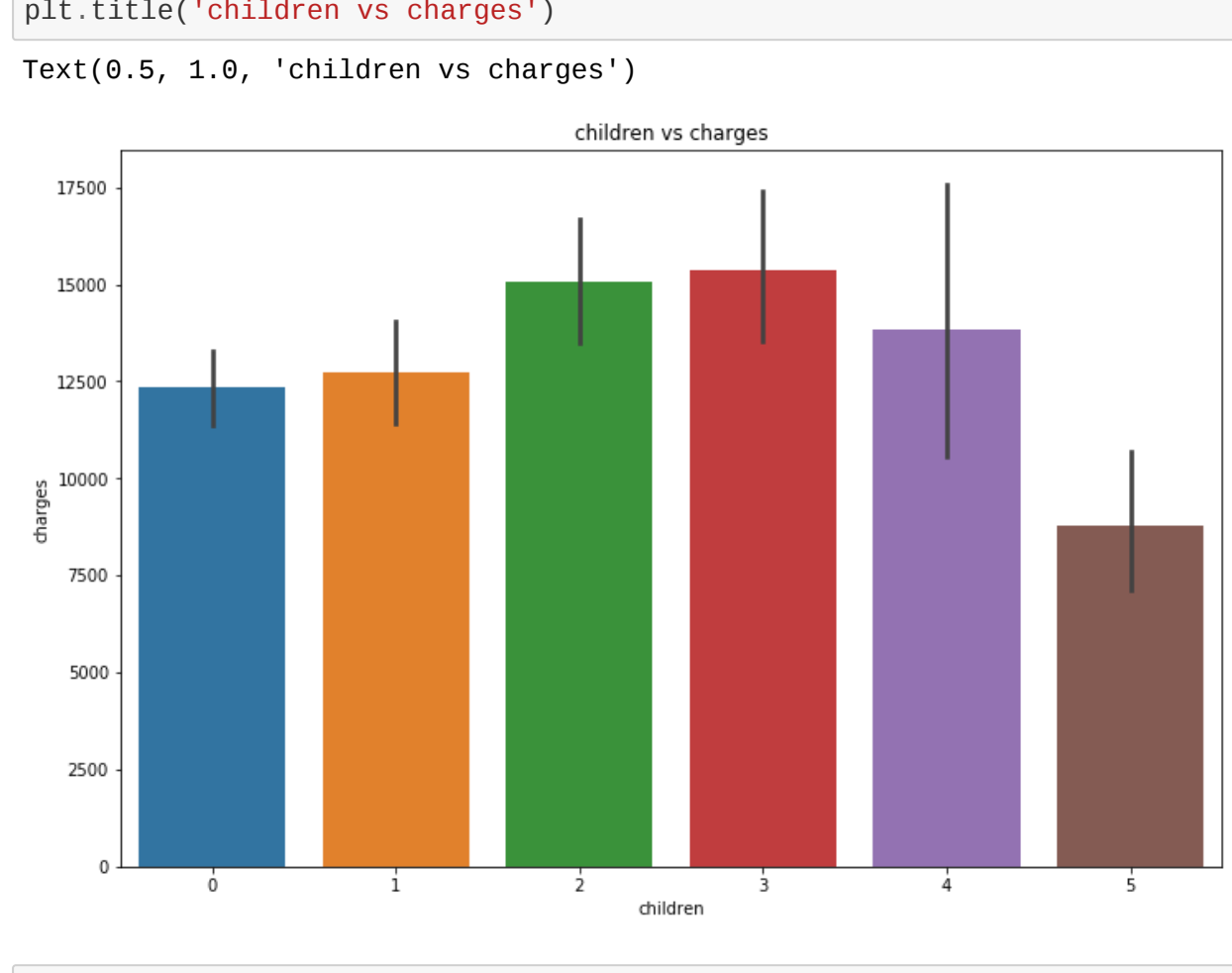
Out[191]: Text(0.5, 1.0, 'sex vs charges')




```
In [192]: plt.figure(figsize = (12, 8))
          sns.barplot(x = 'children', y = 'charges', data = data)

          plt.title('children vs charges')
```

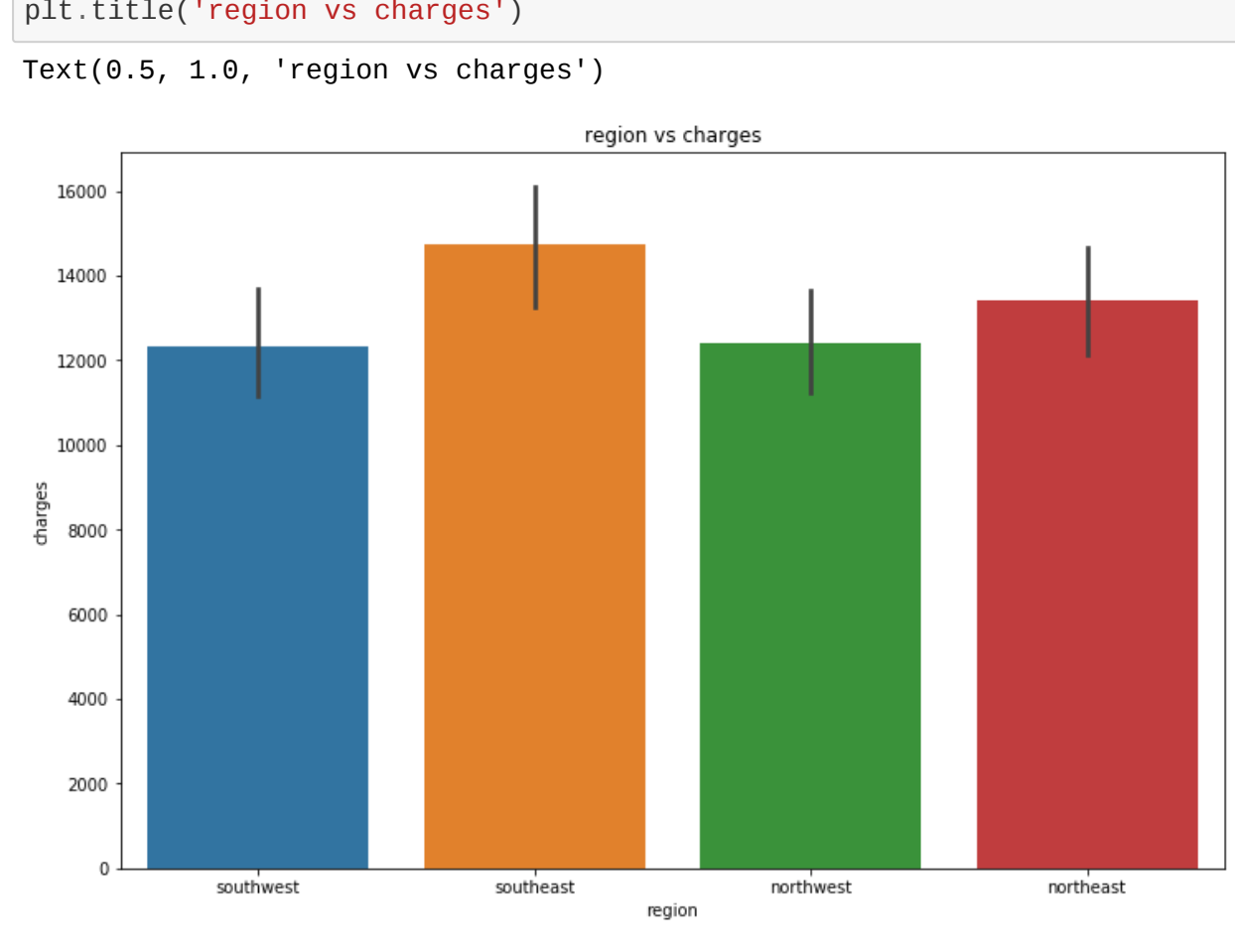
Out[192]: Text(0.5, 1.0, 'children vs charges')




```
In [193]: plt.figure(figsize = (12, 8))
          sns.barplot(x = 'region', y = 'charges', data = data)

          plt.title('region vs charges')
```

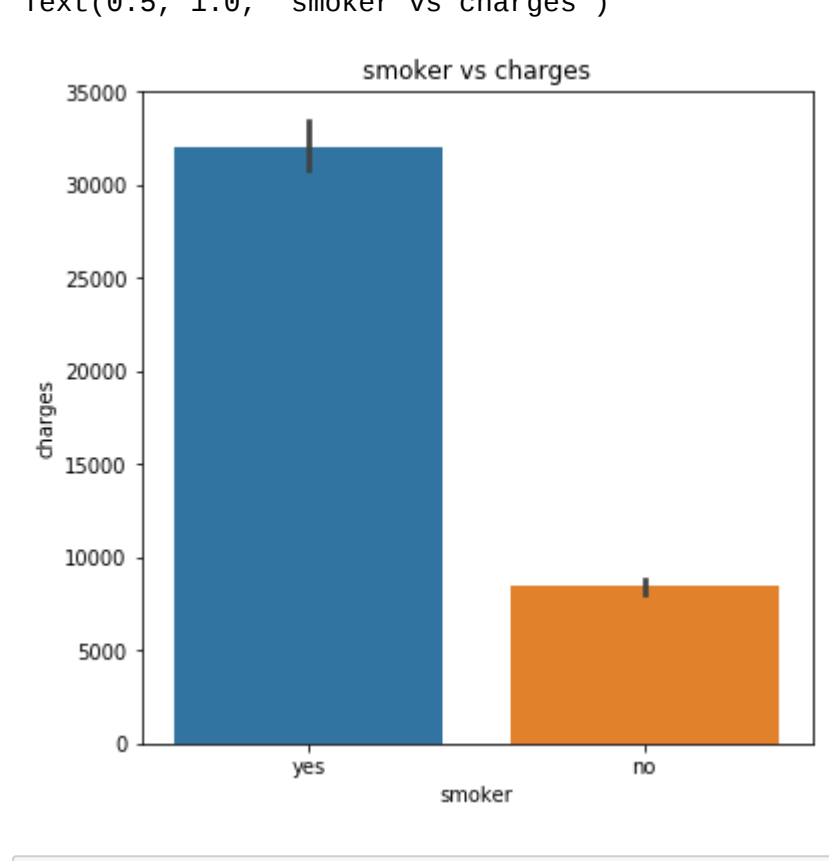
Out[193]: Text(0.5, 1.0, 'region vs charges')




```
In [194]: plt.figure(figsize = (6, 6))
          sns.barplot(x = 'smoker', y = 'charges', data = data)

          plt.title('smoker vs charges')
```

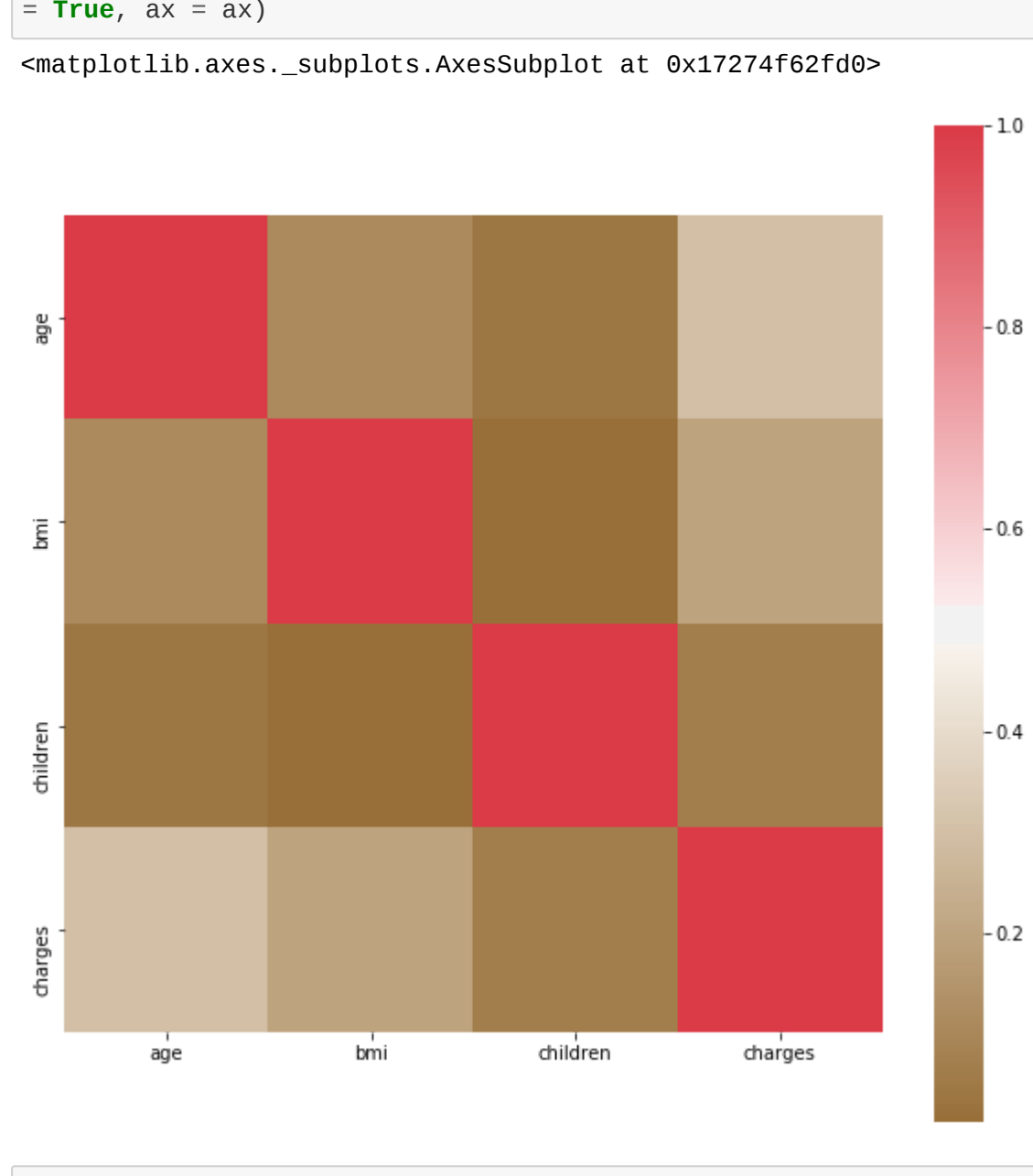
Out[194]: Text(0.5, 1.0, 'smoker vs charges')




```
In [195]: f, ax = plt.subplots(figsize = (10, 10))

          corr = data.corr()
          sns.heatmap(corr, mask = np.zeros_like(corr, dtype = np.bool),
                      cmap = sns.diverging_palette(50, 10, as_cmap = True), square
          = True, ax = ax)
```

Out[195]: <matplotlib.axes._subplots.AxesSubplot at 0x17274f62fd0>




```
In [196]: data = data.drop('region', axis = 1)

          print(data.shape)

          data.columns
```

(1338, 6)

```
Out[196]: Index(['age', 'sex', 'bmi', 'children', 'smoker', 'charges'], dtype='obj
          ect')
```

```
In [197]: from sklearn.preprocessing import LabelEncoder
          le = LabelEncoder()
          data['sex'] = le.fit_transform(data['sex'])
          data['smoker'] = le.fit_transform(data['smoker'])
```

```
In [198]: data['sex'].value_counts()
```

```
Out[198]: 1    676
          0    662
          Name: sex, dtype: int64
```

```
In [199]: data['smoker'].value_counts()
```

```
Out[199]: 0    1064
          1     274
          Name: smoker, dtype: int64
```

```
In [200]: x = data.iloc[:,:5]
          y = data.iloc[:,5]

          print(x.shape)
          print(y.shape)
```

```
(1338, 5)
(1338,)
```

```
In [201]: conda install -c anaconda scikit-learn
```

Note: you may need to restart the kernel to use updated packages.

```
Traceback (most recent call last):
  File "C:\Users\KIIT\Documents\ml\Scripts\conda-script.py", line 11, in
<module>
    from conda.cli import main
ImportError: No module named conda.cli
```

```
In [202]: import sklearn
          print (sklearn.__version__)
```

0.22

```
In [203]: from sklearn.model_selection import train_test_split
          x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.
          2, random_state = 30 )

          print(x_train.shape)
          print(x_test.shape)
          print(y_train.shape)
          print(y_test.shape)
```

```
(1070, 5)
(268, 5)
(1070,)
(268,)
```

```
In [204]: from sklearn.preprocessing import StandardScaler
          sc = StandardScaler()
          x_train = sc.fit_transform(x_train)
          x_test = sc.fit_transform(x_test)
```

```
In [ ]:
```

```
In [262]: from sklearn.linear_model import LinearRegression
          from sklearn.model_selection import train_test_split
          from sklearn.metrics import r2_score
          from keras import metrics
          from sklearn.metrics import log_loss



          lineReg = LinearRegression()
          lineReg.fit(x_train, y_train)
          lineReg.score(x_test, y_test )
          y_pred=lineReg.predict(x_test)



          mse_ln = np.mean((y_test - y_pred)**2, axis = None)
          print("MSE :", mse_ln)


          rmse_ln = np.sqrt(mse_ln)
          print("RMSE :", rmse_ln)


          r2_ln = r2_score(y_test, y_pred)
          print("r2_score :", r2_ln)
          y_score1 = dt.score(x_test, y_pred)
          print('Accuracy: ', y_score1)
```

```
MSE : 37806750.250076555
RMSE : 6148.719399198223
r2_score : 0.7549043319540902
Accuracy:  0.6966217076012765
```

```
In [206]: y_test
```

```
Out[206]: 338    41919.09700
          620     3659.34600
          965     4746.34400
          128    32734.18630
          329     9144.56500
                    ...
          580    12913.99240
          786    12741.16745
          321    24671.66334
          903     8125.78450
          613     6753.03800
          Name: charges, Length: 268, dtype: float64
```

```
In [207]: y_pred
```

```
Out[207]: array([34463.03582081,  5351.63285316,  5380.9763682 , 25960.03740848,
                 12496.7519736 ,  6408.63819349,  4081.62037308,  9300.34454944,
                 32957.30705757, 10937.63104598,  7519.13584334,  9726.30820152,
                 34438.32925237, 11046.23596617,   819.90626006, 35368.51634447,
                 11430.73775682,  8747.50405981, 28520.23066838, 10780.77269818,
                   885.55261833,  8140.91449068,  7686.87521254, 10734.04414171,
                 32605.85490348, 38733.33147903, 14385.6266007 ,  2947.77854767,
                 10456.87417122, 14224.24201886,  1675.51537561, 30193.11285619,
                   944.28091937, 14459.05536274, 12261.67368565,  8411.64973742,
                  1863.86250159,  6487.85043844,   186.61387983,  8192.76868362,
                  6366.87004436, 33513.53920691,  2007.49195393, 13551.24985198,
                 10145.95016308, 29556.16908321,  4878.71697149,  2536.59789849,
                  2321.4068302 ,  4974.23941116, 10987.31701618, 12416.97110239,
                 11958.11247547,   878.6345611 , 10867.31696491,  7014.47673418,
                 30770.43624113,  3085.98932145, 12488.52222358,  4450.51876345,
                 14678.35341681, 10623.53897858, 10948.69439689,  7374.3905582 ,
                  -978.28401768,  7629.31479487,  9810.98011112,  1149.77856134,
                  2548.43516479, 14512.29061742, 10501.54007914,  1841.39577283,
                  -224.99158618,  9265.54108489, 15755.42309527, -1149.84543201,
                 12147.72995061,  9330.21802531, 30356.77333761,  3653.33962314,
                   545.20515877, 10684.17012809,  9022.73780012, 30272.28911391,
                   672.42427126, 16353.28226251,  9013.53855958, 16932.14101336,
                  8763.20999857, 10702.14625251, 13652.63310542, 13267.91285269,
                 26057.54552337, 10462.48053567, 36423.84250946,  5145.79821997,
                  9532.35471109,   926.07280238,  6635.18312192, 34906.78798409,
                 34158.76753135,  8214.44050747,  9411.37431701,  7688.53439131,
                 32529.66951388, 25511.65654536,  6777.39016891, 35454.16845154,
                 33219.62747392,  1609.23659071,  9998.79727626,  6353.29894664,
                   320.81484993, 27468.57195946,  5412.16153686,  3715.05350858,
                 12262.46070132,  9555.31534078,  9525.43925938, 14595.51230853,
                  3855.21657588, 12023.31930406, 35841.13122215, 13831.8140877 ,
                 29096.40483881, 10532.1206343 , 13508.47897596,  8307.15708673,
                 30641.07975475, 11799.75474899, 10913.21723476,  5051.51282042,
                 33133.81423905, 26496.77934186, 11907.79372111,  2425.45213464,
                 40073.30105238, 24746.02998359, 11451.66087289,  6106.9080695 ,
                 12198.53478933,  9957.56176616,  3122.99924398, 36902.66308138,
                  7459.50720854, 10429.20194228,  5980.51463811,  3054.09411045,
                  9315.71342431, 32270.66906763, 15567.01745264,  6190.19131299,
                 10564.69475393,  5912.3169414 ,  8956.52809931,  7866.35712614,
                  4482.86660442, 12337.22896928, 17313.76229093,  4826.20493211,
                 10269.27278997, 30934.25920084, 13836.30990308,  9759.65327348,
                 14733.9054845 , 15013.31001096, 33913.26345748,  7685.12399007,
                  6632.4958633 ,  1590.05645034,  7995.5233211 , 11485.27653018,
                  5580.34296861,  4338.74057296, 32717.41202635,  6957.46627391,
                  2839.55977889,  9807.06898534, 14354.56271204, 29550.1352964 ,
                 10182.87486551, 13822.99846572,  1005.51668237,    72.32794253,
                 38413.17781168, 27229.70114753, 26637.25516294,  6556.18926638,
                 30858.07120572, 14228.50887739,  -840.07324389,  8911.75183635,
                 13619.1000627 , 11509.23467571,  1418.8785819 , 12148.43710257,
                 15333.31465784, 30788.34335414, 29654.44554492,  5250.65574769,
                  9583.20074809,  2499.54931054,  7215.57463317,  6645.54235659,
                 27220.16759396,   814.5900473 ,  4515.0133017 , 36332.3988849 ,
                 34805.58973835, 36453.53611601,  6103.58021016,  6986.1079207 ,
                 38454.74734995,  6306.68110274,  5287.9666742 , 37881.90632257,
                  5323.26143416,  2394.72227414, 16398.81765532, 27936.09993877,
                  9817.33743134, 37684.21361379, 11562.07174432,  1278.94736183,
                  2650.87305617,  9439.79236335,  8875.89112834,  6930.15238323,
                  7026.85741912,  2642.02137428,  9024.07462785, 10720.6914334 ,
                  9706.00665978, 12744.94323341,  9489.66377145, 31193.25443604,
                 29453.69520459, 11057.63638097,  6926.14257954, 27519.87626768,
                 16100.18911202, 14502.11378493,  8747.28306487,  5392.53926128,
                 15805.26112166, 10823.34510868,  1127.65410752,  3976.08097368,
                  4994.98837183, 14941.72586559,  4270.84084452, 10243.5168673 ,
                 13430.53447465, 36555.13522602, 24709.58072542,  3897.41325707,
                 11426.28321239, 10864.09981814, 10704.65146648, 26290.07775662,
                  9177.5825869 , 15075.22243459,  5366.66213128, 11346.31472796,
                 14730.95067629,  5355.25346971, 11734.22431433,  3840.88359743])
```

```
In [253]: from sklearn.svm import SVR
          import numpy
          c_list = numpy.array([])

          for C in c_list:

              sv = SVR(C=C)

              algo = "SVM"


              for name,i_cols_list in X_all:
                  sv.fit(X_train[:,i_cols_list],Y_train)
                  result = mean_absolute_error(numpy.expm1(Y_val), numpy.expm1(mod
          el.predict(X_val[:,i_cols_list])))
                  mae.append(result)
                  print(name + " %s" % result)

              comb.append(algo + " %s" % C )

          sv = SVR()

          sv.fit(x_train, y_train)


          y_pred = sv.predict(x_test)


          mse_sv = np.mean((y_test - y_pred)**2, axis = None)
          print("MSE :", mse_sv)


          rmse_sv = np.sqrt(mse_sv)
          print("RMSE :", rmse_sv)


          r2_sv = r2_score(y_test, y_pred)
          print("r2 score :", r2_sv)
          y_score2 = dt.score(x_test, y_pred)
          print('Accuracy: ', y_score2)
```

```
MSE : 174595630.81989565
RMSE : 13213.46399775228
r2 score : -0.13187810353027452
Accuracy:  -38681.74063389269
```

```
In [225]: from sklearn.ensemble import RandomForestRegressor


          frr = RandomForestRegressor(n_estimators = 40, max_depth = 4, n_jobs = -
          1)


          frr.fit(x_train, y_train)


          y_pred = frr.predict(x_test)

          mse_fr = np.mean((y_test - y_pred)**2, axis = None)
          print("MSE :", mse_fr)


          rmse_fr = np.sqrt(mse_fr)
          print("RMSE :", rmse_fr)


          r2_fr = r2_score(y_test, y_pred)
          print("r2 score :", r2_fr)
          y_score3 = dt.score(x_test, y_pred)
          print('Accuracy: ', y_score3)
          from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegr
          essor
          from sklearn.metrics import make_scorer
          from sklearn.model_selection import cross_val_score
          import timeit
          import math


          def rmsle(real, predicted):
              sum=0.0
              for x in range(len(predicted)):
                  if predicted[x]<0 or real[x]<0: #check for negative values
                      continue
                  p = np.log(predicted[x]+1)
                  r = np.log(real[x]+1)
                  sum = sum + (p - r)**2
              return (sum/len(predicted))**0.5
          y1 = np.random.rand(1000000)
          y2 = np.random.rand(1000000)

          t0 = timeit.default_timer()
          err = rmsle(y1,y2)
          elapsed = timeit.default_timer()-t0
          print('Using loops:')
          print('RMSLE: {:.3f}\nTime: {:.3f} seconds'.format(err, elapsed))
```

```
MSE : 21302505.549231615
RMSE : 4615.463741514131
r2 score : 0.8618989520626663
Accuracy:  0.8733296002989631
Using loops:
RMSLE: 0.280
Time: 13.152 seconds
```

```
In [210]: print(y_pred)
```

```
[44666.34735262  5816.42140988  6006.26409721 16814.29475652
 10713.4290125   6046.05083435  2594.16884896 10909.72221346
 42836.4921955  13104.66885664  6576.64077092 10519.04203692
 45262.39723643 12905.91290024  2646.37643522 25252.37077683
 10519.04203692  9290.88414882 17727.71725536  9844.26855714
  2646.37643522  6555.08507077 10549.25298071  7374.05412917
 25616.37209058 45569.52481809 13669.97585192  5037.85991899
  7112.60901188 11150.73912148  5529.4398407  37307.76092336
  2839.5151206  10983.06229477 13070.4824592   6555.08507077
  2646.37643522  6576.64077092  2646.37643522  6046.3911365
  6649.82710904 23935.60586105  4197.3214711  12994.5648112
  6576.64077092 39618.54648724  6649.82710904  2594.16884896
  4021.68865899  6131.2048962   7171.95101073 10130.905474
 12994.5648112   2646.37643522  7112.60901188  7467.53096498
 20541.37344497  4887.10485067 13070.4824592   6467.00396111
 13045.08790376  6173.36589316 11304.05008322  4941.82833044
  2646.37643522  5764.21382362 11210.23300731  5529.4398407
  6246.88875034 13188.74703218 14483.5812093   2646.37643522
  3956.00800387  9800.50766476 15485.28403881  3956.00800387
  9498.81544912 10130.59710183 19875.37899631  4197.3214711
  2646.37643522  9523.3059137   6046.3911365  18886.29994341
  2646.37643522 14435.66193759  6985.63425521 14644.69514757
 10503.34071168  7098.7725415  14582.51037384 13080.41719837
 36575.1772268   7171.95101073 27088.36896226  4267.65757756
 11164.32073828  6370.76025761  6328.88048189 45869.95313391
 44666.34735262 13104.66885664  7153.72784262  5236.6076013
 39198.2278211  17702.09113143  6610.0403719  44666.34735262
 25741.90892867  2839.5151206  13190.52124381  7281.21420947
  6719.17281584 18518.04803379  4267.65757756  6170.0571377
 12994.5648112   6555.08507077  8444.58703726 10021.7555751
  3759.20735301  7107.73479305 26459.97678989 13502.91332894
 18886.29994341 14277.93060185 14373.47716386  7226.90631185
 23631.95383835 12905.91290024 10940.65082066  4907.6916119
 40611.38169243 36414.77640553 12994.5648112   2646.37643522
 46085.87514111 18261.42446616 13780.07989737  6396.42709474
 12905.91290024 10955.63448249  5961.01447447 45885.49822893
  6792.82926731 10894.73855163  5349.05288744  6780.55952769
  7226.90631185 25530.49788502 13045.08790376  9725.8354198
  6173.36589316  7413.60210117  6576.64077092  6752.71465972
  2594.16884896 12994.5648112  13416.13186641  2594.16884896
 10099.80138424 38614.76626091 10759.34128153 10116.95367735
 10782.82716449 13502.91332894 41955.5038993   6328.88048189
  5929.43248457  4415.89675237  7040.58955634 12905.91290024
  7033.45392044  5816.42140988 39083.91934904  9432.82800639
  2594.16884896  4791.07326212 14373.47716386 19875.37899631
 10212.93967053 13045.08790376  2646.37643522  2646.37643522
 45885.49822893 39170.03153101 16814.29475652  6649.82710904
 38608.62545424 10713.4290125   2646.37643522  5236.6076013
 14167.82655641 11243.15415236  2646.37643522 12994.5648112
 15485.28403881 27087.29826394 38137.51443193  5184.53299064
  6985.63425521  2594.16884896  6576.64077092  5453.63782023
 38381.38187464  6515.3533222   2787.30753433 45475.3259626
 42382.81658157 42701.11016678  6484.97957343  6610.0403719
 45991.67628561  9725.8354198   4941.82833044 46017.24273637
  4267.65757756  2646.37643522 13500.55588539 16986.29065653
  6985.63425521 45569.52481809  7273.13464268  2646.37643522
  5003.72320045  7226.90631185  6576.64077092  4791.07326212
  6762.95752643  2594.16884896  7153.72784262 13016.01694569
 10116.95367735 12994.5648112  10986.51199407 23467.15617407
 39618.54648724 12905.91290024  6752.71465972 36614.15965681
 14722.36765588 14980.68597639 10549.25298071  2594.16884896
 15069.09716984 14277.93060185  2646.37643522  2787.30753433
  4678.06218744 13188.74703218  6648.18304671 10185.91269146
 10977.8964186  26827.59545562 19916.49796771  6574.57975125
 11120.4927928  10714.11129102 13104.66885664 20328.44769091
  6576.64077092 14167.82655641  4791.07326212 13416.47770986
 13264.66468017  6467.00396111 10518.3597584   6850.66502503]
```

```
In [211]: print(y_test)
```

```
338    41919.09700
620     3659.34600
965     4746.34400
128    32734.18630
329     9144.56500
          ...
580    12913.99240
786    12741.16745
321    24671.66334
903     8125.78450
613     6753.03800
Name: charges, Length: 268, dtype: float64
```

```
In [212]: from sklearn.tree import DecisionTreeRegressor
          from sklearn.metrics import r2_score


          dt = DecisionTreeRegressor()


          dt.fit(x_train, y_train)


          y_pred = dt.predict(x_test)


          mse_dtr = np.mean((y_test - y_pred)**2, axis = None)
          print("MSE :", mse_dtr)


          rmse_dtr = np.sqrt(mse_dtr)
          print("RMSE :", rmse_dtr)


          r2_dtr = r2_score(y_test, y_pred)
          print("r2 score :", r2_dtr)

          y_score4 = dt.score(x_test, y_pred)
          print('Accuracy: ', y_score4)
```

```
MSE : 39115165.72528189
RMSE : 6254.211838855627
r2 score : 0.7464220645585631
Accuracy:  1.0
```

```
In [213]: from sklearn.preprocessing import PolynomialFeatures
          poly_regs= PolynomialFeatures(degree= 2)
          x_poly= poly_regs.fit_transform(x)
          lin_reg_2 =LinearRegression()
          lin_reg_2.fit(x_poly, y)
```

```
Out[213]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize
          =False)
```

```
In [ ]:
```

```
In [214]: from sklearn.preprocessing import PolynomialFeatures
          from sklearn.metrics import mean_squared_error
          from sklearn import metrics
          polynomial_features= PolynomialFeatures(degree=2)
          x_poly = polynomial_features.fit_transform(x)
          model = LinearRegression()
          model.fit(x_poly, y)
          y_poly_pred = model.predict(x_poly)
          mse = metrics.mean_squared_error(y,y_poly_pred)
          r2 = r2_score(y,y_poly_pred)
          rmse = np.sqrt(mean_squared_error(y,y_poly_pred))

          print('MSE:',mse)

          print('RMSE for Polynomial Regression=>',rmse)
          y_score5 = dt.score(x_test, y_pred)
          print('Accuracy: ', y_score5)
          print(r2)
```

```
MSE: 22866274.364422705
RMSE for Polynomial Regression=> 4781.869337865967
Accuracy:  1.0
0.8439617668509795
```

```
In [215]: random_seed = 12
```

```
In [216]: compare_models = pd.DataFrame(
              {  'Model' : ['LineReg', 'dt','frr','sv','model'],
                 'Score' : [r2_ln, r2_dtr,r2_fr,r2_sv,r2],
                  'MSE'  : [mse_ln, mse_dtr,mse_fr, mse_sv,mse ],
                  'RMSE'  : [rmse_ln, rmse_dtr,rmse_fr,rmse_sv,rmse] ,
               'Accuracy' :[y_score1,y_score4,y_score3,y_score2,y_score5]
              })

          print(compare_models)
```

```
     Model     Score           MSE          RMSE      Accuracy
0  LineReg  0.754904  3.780675e+07   6148.719399      0.708841
1       dt  0.746422  3.911517e+07   6254.211839      1.000000
2      frr  0.862533  2.120476e+07   4604.862293      0.884227
3       sv -0.131878  1.745956e+08  13213.463998 -38248.616735
4    model  0.843962  2.286627e+07   4781.869338      1.000000
```

```
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
In [ ]:
```