

main__auto

October 17, 2023

1 Advanced Java & Advanced Python Assignment

1.1 Deng Chuan Chang | Yasser El Karkouri | Julien Godfroy

1.1.1 Advertising Dataset

```
[ ]: import matplotlib.pyplot as plt
      from scipy.stats import spearmanr

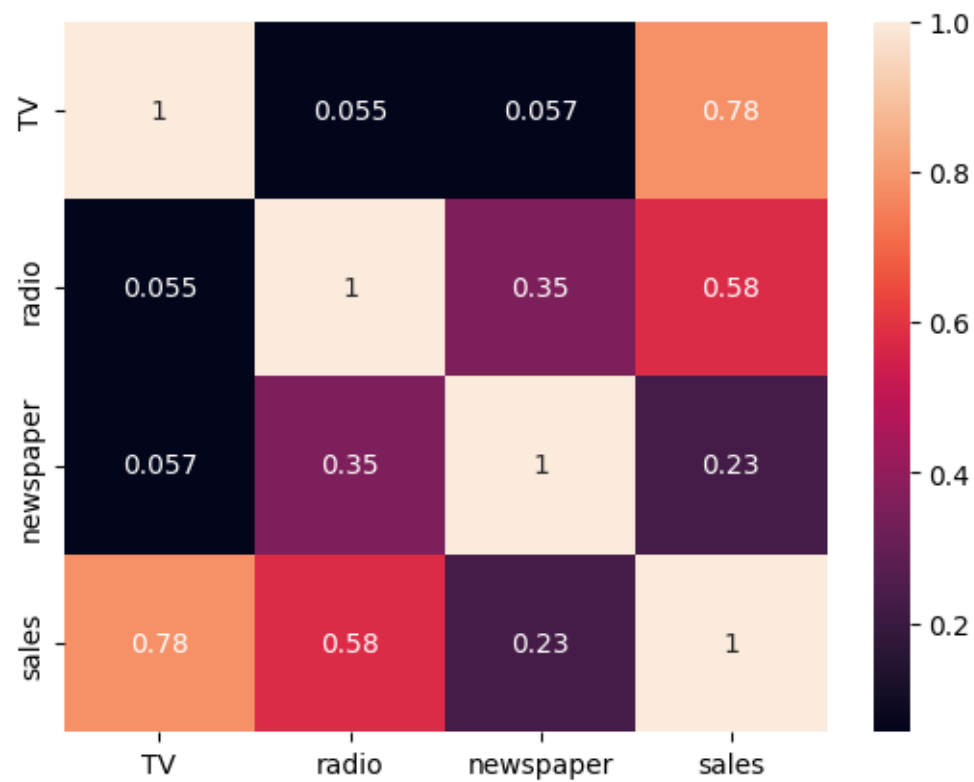
      from Class.ModelClass import *      # Importing the Model class from ModelClass.
      ↪py
      from functions.utils import *      # Importing the utils functions from utils.py

      ### Before computing regression, let's do some data analysis

[ ]: x, y, df = import_clean_data('./data/Advertising.csv', input_list=['TV', '
      ↪radio', 'newspaper'], output_list=['sales'])
      df.head()

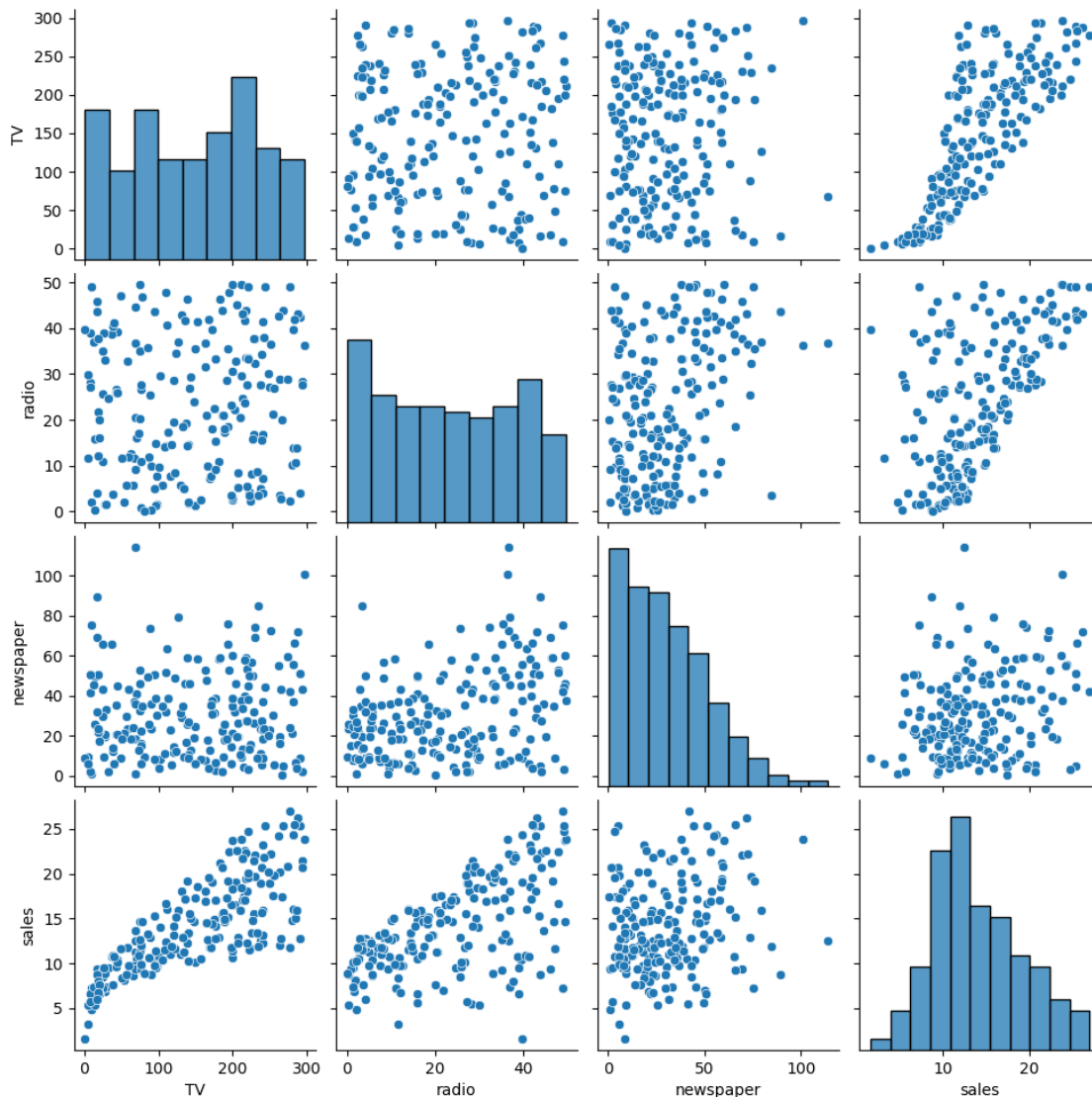
      #heatmap for the correlation coefficient between the variables
      import seaborn as sns
      sns.heatmap(df.corr(), annot=True)

[ ]: <AxesSubplot: >
```



```
[ ]: #plot correlation between all variables
      %matplotlib inline
      sns.pairplot(df)
```

```
[ ]: <seaborn.axisgrid.PairGrid at 0x2cef6eb14c0>
```



```
[ ]: #spearman correlation for all combination of columns
for i in range(len(df.columns)):
    for j in range(i+1, len(df.columns)):
        corr, _ = spearmanr(df[df.columns[i]], df[df.columns[j]])
        print('Spearman correlation between {} and {} is: {}'.format(df.
↪columns[i], df.columns[j], corr))
```

Spearman correlation between TV and radio is: 0.05612339226247207
 Spearman correlation between TV and newspaper is: 0.05083973485105542
 Spearman correlation between TV and sales is: 0.8006143768505688
 Spearman correlation between radio and newspaper is: 0.3169794890663236
 Spearman correlation between radio and sales is: 0.5543037314053145
 Spearman correlation between newspaper and sales is: 0.19492188424873094

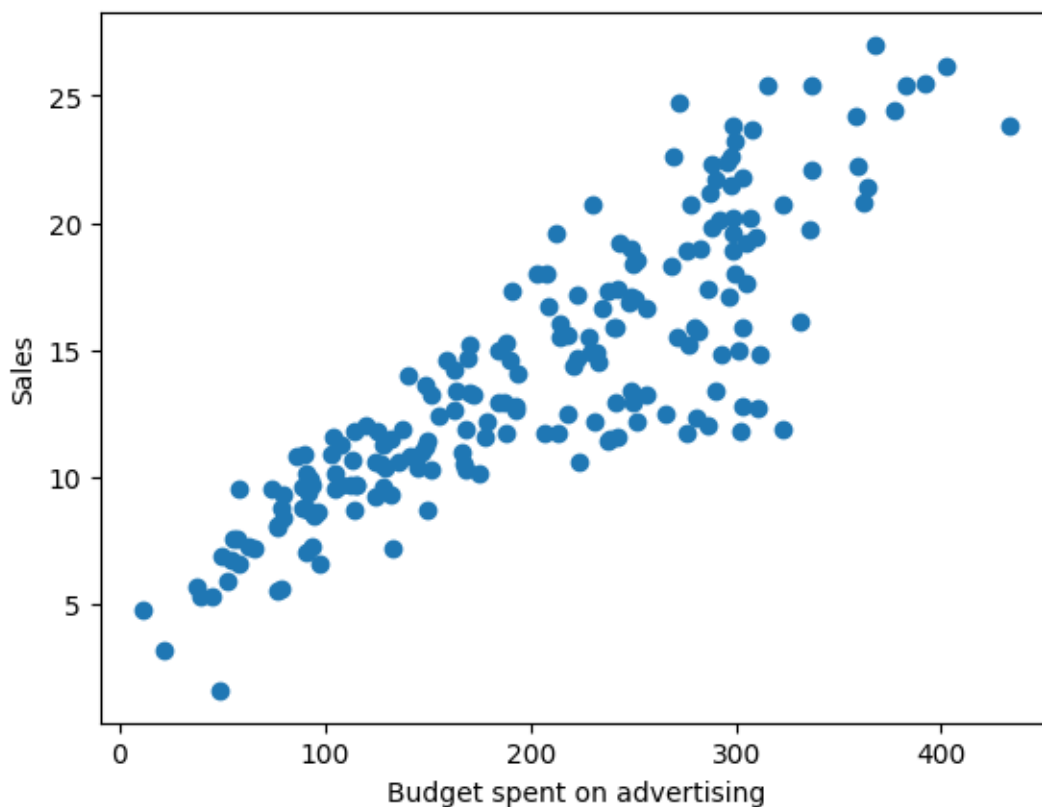
```
[ ]: # Create a new column of the sum of the budget spent on advertising
df['adv_budget'] = df['TV'] + df['radio'] + df['newspaper']

#spearman
corr, _ = spearmanr(df['adv_budget'], df['sales'])
print('Spearman correlation between sum and sales is: {}'.format(corr))

#plot
plt.scatter(df['adv_budget'], df['sales'])
plt.xlabel('Budget spent on advertising')
plt.ylabel('Sales')
```

Spearman correlation between sum and sales is: 0.8770508999294694

```
[ ]: Text(0, 0.5, 'Sales')
```



Let's compute the regression using TV and RADIO predictors only

```
[ ]: x, y, df = import_clean_data('./data/Advertising.csv', input_list=['TV', 'radio'],
    ↪ output_list=['sales'])
X, y = prepare_vectors(x, y)
```

Let's find out what is the best combinaison of: - Test size - Number of iterations - Learning rate

```
[ ]: test_size_list = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7]
iteration_list = [50, 100, 500, 1000, 3000, 5000, 7000, 10000]
rate_list = [0.1, 0.05, 0.01, 0.001, 0.0001]

model1_df, model1_dict = find_combination(X, y, test_size_list, iteration_list,
↪rate_list)
```

```
[ ]: model1_df
#model1_df.to_csv('model_df.csv') #to save in a csv file

model1_df.head()
```

```
[ ]:      test_size  iteration  rate  r_square      mse
0         0.4      10000.0  0.10  0.914199  2.232220
1         0.4       7000.0  0.10  0.914199  2.232220
2         0.4      10000.0  0.05  0.914199  2.232220
3         0.4       5000.0  0.10  0.914199  2.232220
4         0.4       7000.0  0.05  0.914197  2.232263
```

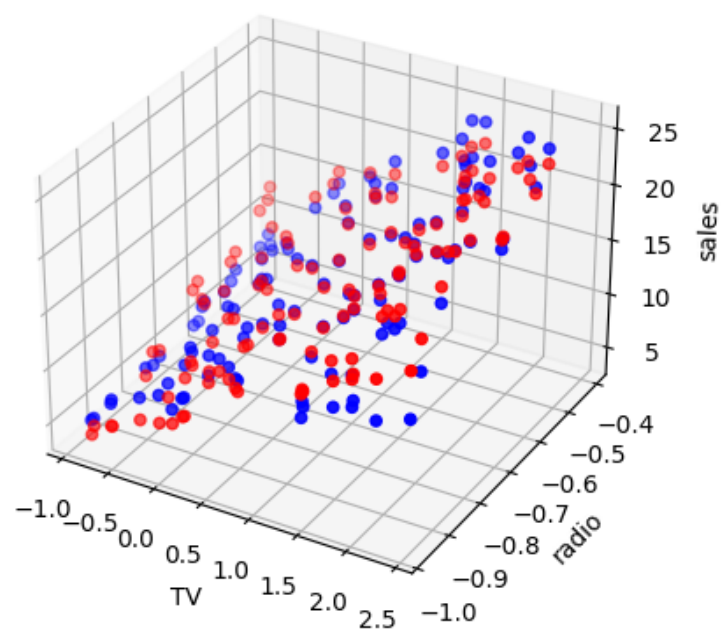
In this case, the best model the 3rd one. Because the other computes more iterations without improving significativly the model : Best : - Test size = 0.4 - Number of iterations = 5000 - Learning rate = 0.1 Warning : because of the randomness of the split, the best model can change from one execution to another

Let's visualize the regression line and the cost function

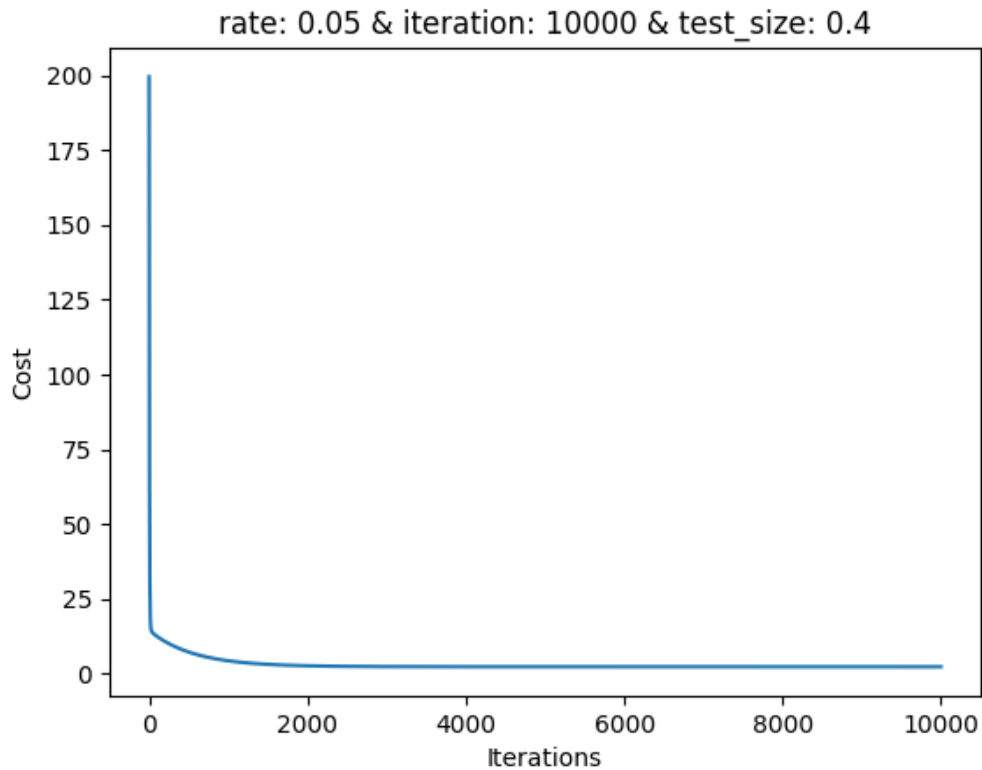
```
[ ]: best_model = model1_dict[2][1] #get best model (the 4th one)
#make sure ipympl is installed (pip intall ipympl)
%matplotlib widget
best_model.plot_regression_3D('TV', 'radio', 'sales')
best_model.theta
```

```
[ ]: array([[ 3.88722432],
          [16.1534349 ],
          [22.87372515]])
```

Regression : Red & Data : Blue



```
[ ]: %matplotlib widget
      best_model.plot_cost()
```



Now we can test our model

```
[ ]: mse, r_square, predictions = best_model.test_model()
      print('mse: ', mse)
      print('r_square: ', r_square)
```

```
mse: 3.6984672078776635
r_square: 0.8685643402626058
```

1.1.2 Let's import the data Advertising.csv to process TV and RADIO and NEWSPAPERS predictors with the same methodology

Let's find out what is the best combinaison of: - Test size - Number of iterations - Learning rate

```
[ ]: #import data
      x, y, df = import_clean_data('./data/Advertising.csv', input_list=['TV',
      ↪ 'radio', 'newspaper'], output_list=['sales'])
      X, y = prepare_vectors(x, y)

[ ]: #compute all models
      model2_df, model2_dict = find_combination(X, y, test_size_list, iteration_list,
      ↪ rate_list)
```

```
[ ]: model2_df.head()
```

```
[ ]:      test_size  iteration  rate  r_square      mse
0         0.4    10000.0  0.10  0.915685  2.193546
1         0.4     7000.0  0.10  0.915685  2.193546
2         0.4     5000.0  0.10  0.915685  2.193546
3         0.4    10000.0  0.05  0.915685  2.193546
4         0.4     7000.0  0.05  0.915685  2.193549
```

```
[ ]: #Test of the model
model2_best = model2_dict[2][1] #get best model (the 3st one)
mse, r_square, predictions = model2_best.test_model()
print('mse: ', mse)
print('r_square: ', r_square)
```

mse: 3.862022758723319

r_square: 0.8627519237882015

Create a 3D plot and add a colorbar which maps values to colors to represent the sales

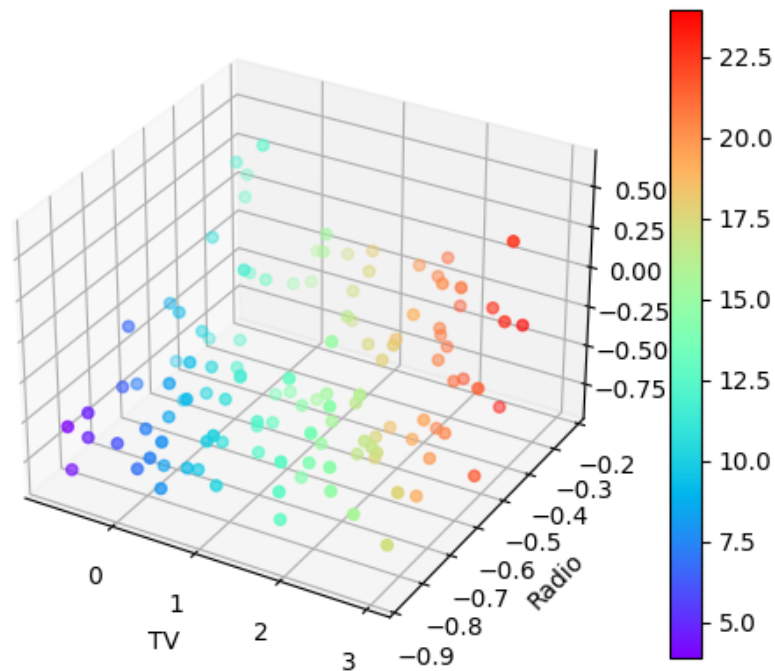
```
[ ]: #Create a 3D plot and add a colorbar which maps values to colors to represent
      ↳ the sales
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Utiliser la couleur pour représenter les ventes
scatter = ax.scatter(model2_best.X_train[:,0], model2_best.X_train[:,1],
      ↳ model2_best.X_train[:,2], c=model2_best.X_train.dot(model2_best.theta),
      ↳ cmap='rainbow')

# Ajouter une barre de couleur
plt.colorbar(scatter)

# Ajouter des étiquettes d'axes
ax.set_xlabel('TV')
ax.set_ylabel('Radio')
ax.set_zlabel('Newspaper')

# Afficher le graphique
plt.show()
```

Suppose that spending money on radio advertising actually increases the effectiveness of TV advertising

$\text{sales} = 0 + 1 * \text{TV} + 2 * \text{radio} + 3 * (\text{radio} * \text{TV})$

```
[ ]: x, y, df = import_clean_data('./data/Advertising.csv', input_list=['TV',
    ↪ 'radio', 'newspaper'], output_list=['sales'])
X, y = prepare_vectors(x, y)

X[:,2] = X[:,0] * X[:,1] #add interaction between TV and radio and rewrite it
    ↪ in the 3rd column

model3_df, model3_dict = find_combination(X, y, test_size_list, iteration_list,
    ↪ rate_list)
```

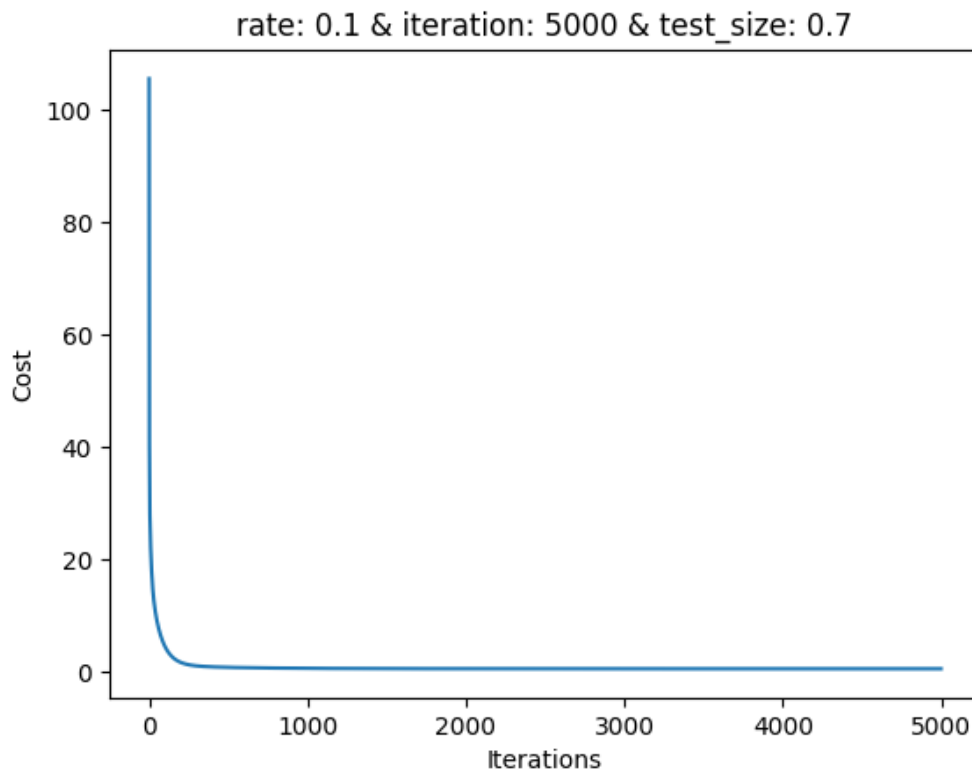
```
[ ]: model3_df.head(5)
#model3_df.to_csv('model3_df.csv') #to save in a csv file
```

```
[ ]:      test_size  iteration  rate  r_square      mse
0         0.7      10000.0  0.10  0.978194  0.611915
1         0.7       7000.0  0.10  0.978194  0.611915
2         0.7      10000.0  0.05  0.978194  0.611922
3         0.7       5000.0  0.10  0.978194  0.611924
4         0.7       7000.0  0.05  0.978187  0.612110
```

In this case, the best one is the 4th one. Because the other computes more iterations without improving the model.

Best : - Test size = 0.7 - Number of iterations = 5000 - Learning rate = 0.1

```
[ ]: best_model3 = model3_dict[3][1] #get best model (the 1st one)
      %matplotlib widget
      best_model3.plot_cost()
```



```
[ ]: #test the model with the test set
      mse, r_square, predictions = best_model3.test_model()
      print('mse: ', mse)
      print('r_square: ', r_square)
```

```
mse: 1.0150663307880718
r_square: 0.9616161673015073
```

```
[ ]: # 3D plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Color to represent sales
scatter = ax.scatter(best_model3.X_train[:,0], best_model3.X_train[:,1],
                    ↪best_model3.X_train[:,2], c=best_model3.X_train.dot(best_model3.theta),
                    ↪cmap='rainbow')
plt.colorbar(scatter)

ax.set_xlabel('TV')
ax.set_ylabel('Radio')
ax.set_zlabel('Advertising Budget')

plt.show()
```

