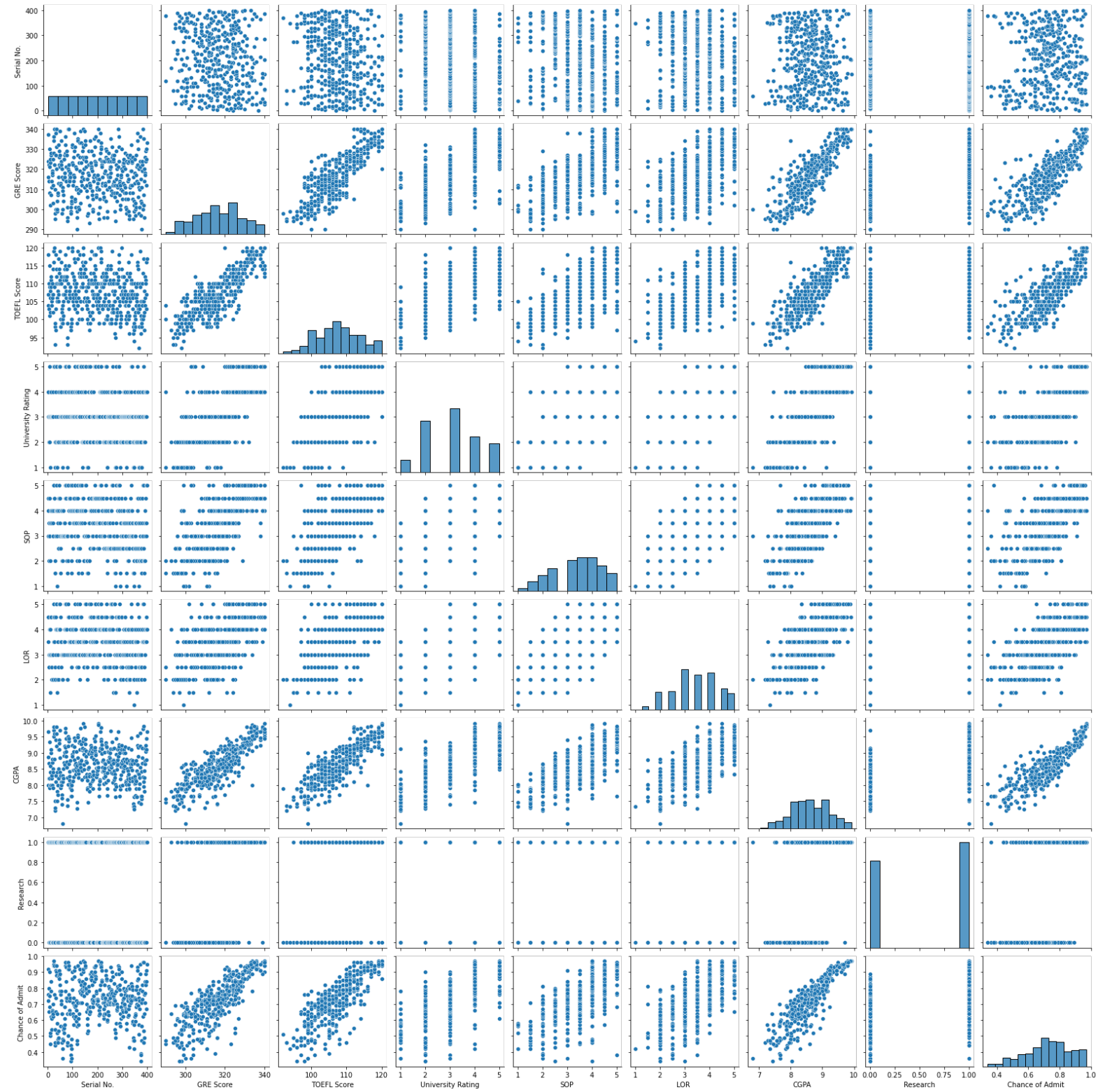
In [18]:

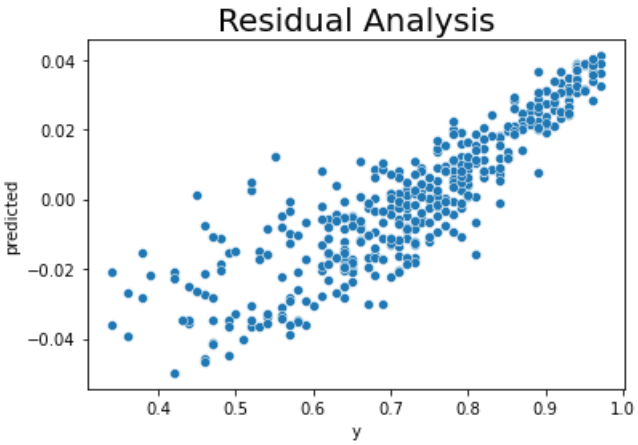
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
G_admission = pd.read_csv("D:\\archive\\Admission_Predict.csv", sep=",", index_col=False)
G_admission.head()
#plot the data
sns.pairplot(data= G_admission, kind="scatter")
plt.show()
print(G_admission.corr())
#check null values
Null = G_admission.isnull()
Null.sum()
#drops irrelavent columns
G_admission = G_admission.drop([G_admission.columns[0],'Research'], axis=1)
#split data into dependent and independent
x = G_admission.values[:, 0:6]
y = G_admission.values[:, 6]
m = len(y)
print('Total no of training examples (m) = %s \n' %(m))
#data normalization
def normalize(X):
    mu = np.mean(x, axis = 0)
    sigma = np.std(x, axis= 0, ddof = 1)
    X_normalize = (x - mu)/sigma
    return X_normalize, mu, sigma
x, mu, sigma = normalize(x)
print('X_normalize= ', x[:5])
mu_testing = np.mean(x, axis = 0) # mean
sigma_testing = np.std(x, axis = 0, ddof = 1)
def compute_cost(x, y, theta):
     predictionsOfy = x.dot(theta)
     errors = np.subtract(predictionsOfy, y)
     J = 1/(m) * errors.T.dot(errors)
     return J, predictionsOfy
def gradient_descent(x, y, theta, alpha, iterations):
    cost_history = np.zeros(iterations)
    for i in range(iterations):
      predictions = x.dot(theta)
      errors = np.subtract(predictions, y)
      sum_delta = (alpha / m) * x.transpose().dot(errors)
      theta = theta - sum_delta
      cost_history[i], predictionsOfy = compute_cost(x, y, theta)
    return theta, cost_history, predictionsOfy
theta = np.zeros(6)
iterations = 400;
alpha = 0.0001;
theta, cost_history, predictionsOfy = gradient_descent(x, y, theta, alpha, iterations)
print('First 5 values from cost_history =', cost_history[:5])
print('Last 5 values from cost_history =', cost_history[-5 :])
#residual plot
sns.scatterplot(x=y ,y=predictionsOfy)
plt.xlabel('y')
plt.ylabel('predicted')
plt.title("Residual Analysis",fontsize=20)
plt.show()
#seeing the impact of changing learning rate
alpha = 0.001;
theta_1, cost_history_1, predictionsOfy = gradient_descent(x, y, theta, alpha, iterations)
alpha = 0.01;
theta_2, cost_history_2, predictionsOfy = gradient_descent(x, y, theta, alpha, iterations)
alpha = 0.1;
theta_3, cost_history_3, predictionsOfy = gradient_descent(x, y, theta, alpha, iterations)
alpha = 1;
theta_4, cost_history_4, predictionsOfy = gradient_descent(x, y, theta, alpha, iterations)
alpha = 10;
theta_5, cost_history_5, predictionsOfy = gradient_descent(x, y, theta, alpha, iterations)
alpha = 100;
theta_6, cost_history_6, predictionsOfy = gradient_descent(x, y, theta, alpha, iterations)
print('Seeing the impact of changing learning rate')
plt.plot(range(1, iterations +1), cost_history, color ='brown', label = 'alpha =0.0001')
plt.show()

plt.plot(range(1, iterations +1), cost_history_1, color ='purple', label = 'alpha = 0.001')
plt.show()
plt.plot(range(1, iterations +1), cost_history_2, color ='red', label = 'alpha = 0.01')
```

```
plt.show()
plt.plot(range(1, iterations +1), cost_history_3, color ='green', label = 'alpha = 0.1')
plt.show()
plt.plot(range(1, iterations +1), cost_history_4, color ='yellow', label = 'alpha = 1')
plt.show()
plt.plot(range(1, iterations +1), cost_history_5, color ='blue', label = 'alpha = 10')
plt.show()
plt.plot(range(1, iterations +1), cost_history_6, color ='black', label = 'alpha = 100')
plt.show()

#linear model using sklern
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=9,test_size=0.25)
model= LinearRegression()
model.fit(x_train,y_train)
pred = model.predict(x_test)
print("MSE :",mean_squared_error(y_test,pred))
sns.scatterplot(x=y_test ,y=pred)
plt.xlabel('y')
plt.ylabel('predicted')
plt.title("Residual Analysis",fontsize=20)
plt.show()
```

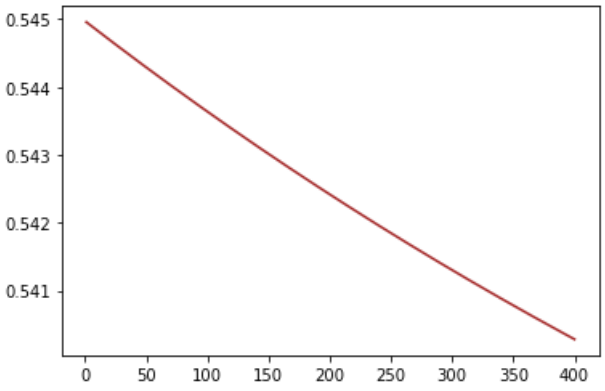


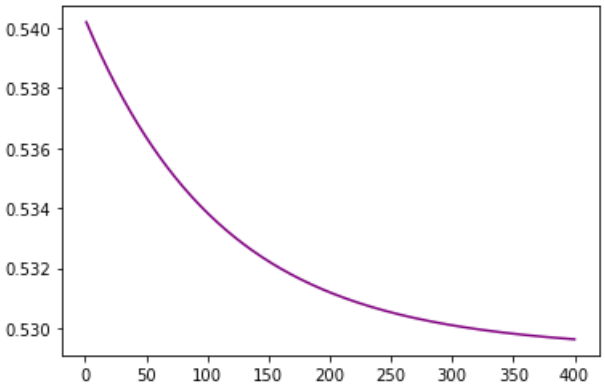

```
              Serial No.  GRE Score  TOEFL Score  University Rating  \
Serial No.      1.000000  -0.097526    -0.147932          -0.169948
GRE Score      -0.097526   1.000000     0.835977           0.668976
```

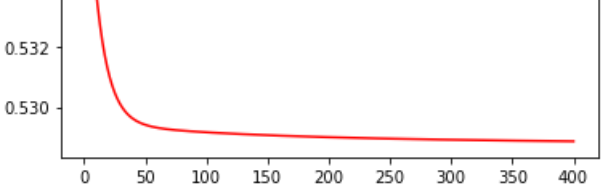
```
TOEFL Score         -0.147932   0.835977     1.000000          0.695590
University Rating   -0.169948   0.668976     0.695590          1.000000
SOP                 -0.166932   0.612831     0.657981          0.734523
LOR                 -0.088221   0.557555     0.567721          0.660123
CGPA                -0.045608   0.833060     0.828417          0.746479
Research            -0.063138   0.580391     0.489858          0.447783
Chance of Admit      0.042336   0.802610     0.791594          0.711250

                    SOP       LOR       CGPA  Research  Chance of Admit
Serial No.      -0.166932 -0.088221 -0.045608 -0.063138         0.042336
GRE Score        0.612831  0.557555  0.833060  0.580391         0.802610
TOEFL Score      0.657981  0.567721  0.828417  0.489858         0.791594
University Rating  0.734523  0.660123  0.746479  0.447783       0.711250
SOP              1.000000  0.729593  0.718144  0.444029         0.675732
LOR              0.729593  1.000000  0.670211  0.396859         0.669889
CGPA             0.718144  0.670211  1.000000  0.521654         0.873289
Research         0.444029  0.396859  0.521654  1.000000         0.553202
Chance of Admit  0.675732  0.669889  0.873289  0.553202         1.000000
Total no of training examples (m) = 400

X_normalize=  [[ 1.75990263  1.74478556  0.79782946  1.09249604  1.16586107  1.76261088]
 [ 0.62687135 -0.06755072  0.79782946  0.59590693  1.16586107  0.45458197]
 [-0.07037867 -0.56182425 -0.0765042  -0.39727129  0.05286721 -1.00437335]
 [ 0.45255884  0.42672281 -0.0765042   0.09931782 -1.06012666  0.11918994]
 [-0.24469118 -0.72658209 -0.95083785 -1.3904495  -0.50362972 -0.65221172]]
First 5 values from cost_history = [0.54495556 0.54494164 0.54492772 0.54491382 0.54489993]
Last 5 values from cost_history = [0.54032183 0.54031206 0.5403023  0.54029255 0.5402828 ]
```

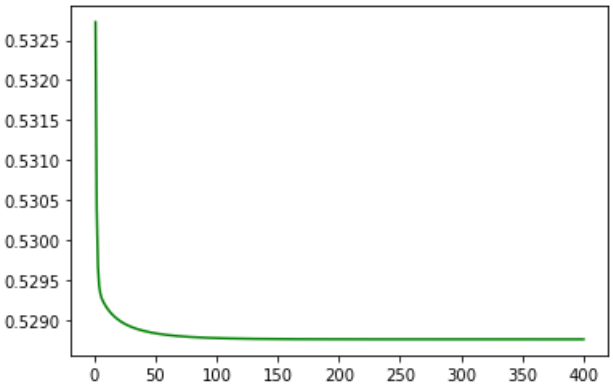


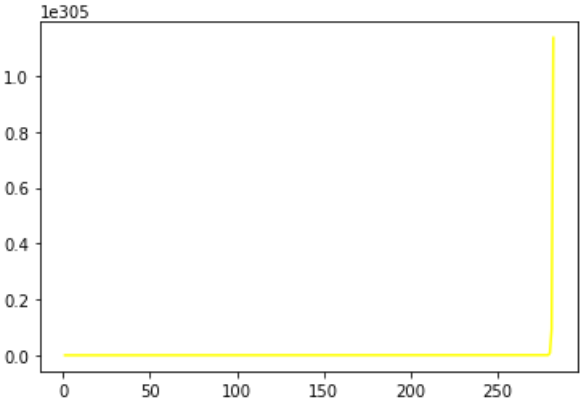
Seeing the impact of changing learning rate

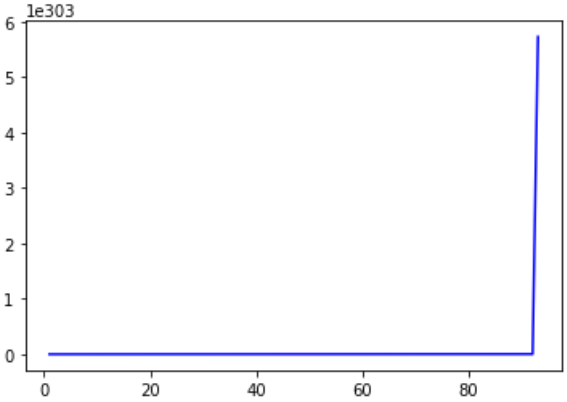


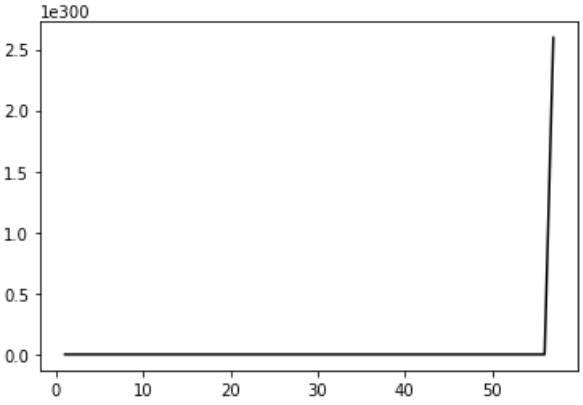




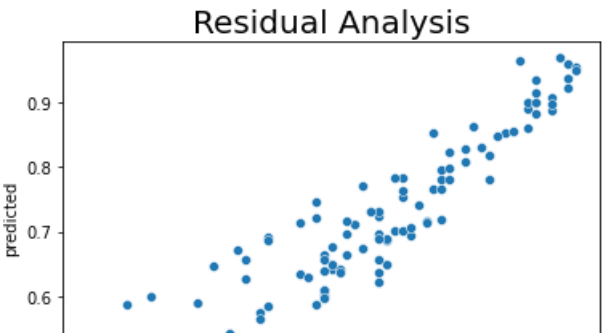


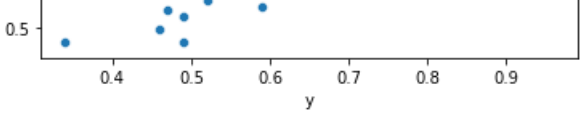






MSE : 0.0033518802603911863





In [ ]:

In [ ]: