In [67]:

```
import matplotlib.pyplot as plt
import operator
import numpy as np
import pandas as pd
import io
import os
import math
import random
import statistics
import itertools
from scipy.stats import pearsonr
from sklearn.utils import shuffle
from sklearn.preprocessing import  StandardScaler
from sklearn.linear_model import LogisticRegression
from google.colab import drive
drive.mount('/content/gdrive/')
```

```
Drive already mounted at /content/gdrive/; to attempt to forcibly remount, call
drive.mount("/content/gdrive/", force_remount=True).
```

In [68]:

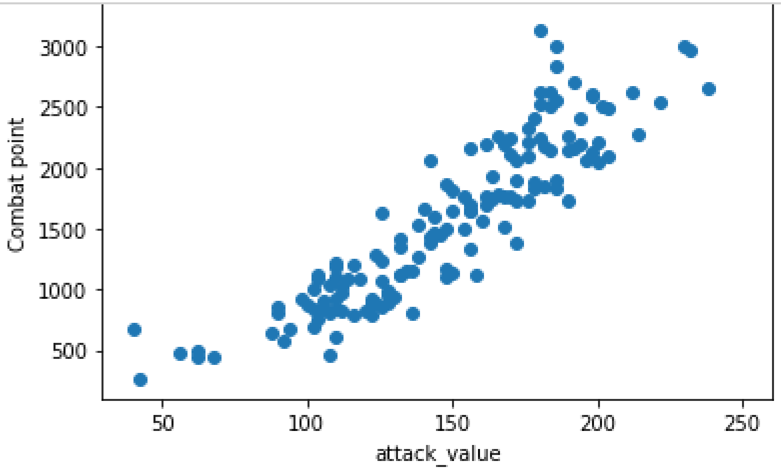
```

trainingdata = pd.read_csv('/content/gdrive/MyDrive/Data/hw2_data.csv')
inputdata = trainingdata[trainingdata.columns[1:-1]]
attributes = trainingdata.columns[1:8]
outputdata = trainingdata.columns[-1:]

print(outputdata)
print(trainingdata)
```

```
Index(['combat_point'], dtype='object')
           name  stamina  ...  primary_strength  combat_point
0     Bulbasaur       90  ...             Grass          1079
1       Ivysaur      120  ...             Grass          1643
2      Venusaur      160  ...             Grass          2598
3    Charmander       78  ...              Fire           962
4    Charmeleon      116  ...              Fire          1568
..          ...      ...  ...               ...           ...
141  Aerodactyl      160  ...              Rock          2180
142     Snorlax      320  ...            Normal          3135
143     Dratini       82  ...            Dragon           990
144   Dragonair      122  ...            Dragon          1760
145   Dragonite      182  ...            Dragon          3525

[146 rows x 9 columns]
```

In [69]:
```
colmns = inputdata.columns
numeric_colmns = inputdata._get_numeric_data().columns
print(numeric_colmns)
categoric_colmns = pd.DataFrame(list(set(colmns) - set(numeric_colmns)))
print(list(set(colmns) - set(numeric_colmns)))
print(list(set(numeric_colmns)))
print(categoric_colmns)
```

```
Index(['stamina', 'attack_value', 'defense_value', 'capture_rate', 'flee_rate',
       'spawn_chance'],
      dtype='object')
['primary_strength']
['defense_value', 'stamina', 'flee_rate', 'spawn_chance', 'attack_value', 'capt
ure_rate']
                  0
0  primary_strength
```

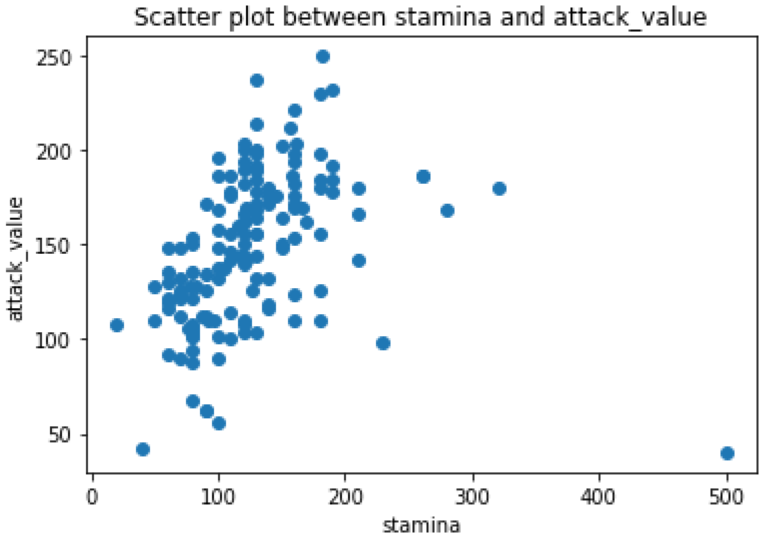
In [70]:
```
#2.ii


columns1 = {'stamina', 'attack_value','defense_value','capture_rate','flee_rate',
for i in columns1:
  plt.scatter(trainingdata[i],trainingdata.combat_point)
  plt.title("Scatter plot between " +i+ " and Combat point")
  plt.xlabel(i)
  plt.ylabel("Combat point")
  plt.tight_layout
  plt.show()
  r,p= pearsonr(trainingdata[i],trainingdata.combat_point)
  print("Pearson's correlation between " +i+ " and combat point is " + format(r))
```



```
Pearson's correlation between attack_value and combat point is 0.907531540104
2733
```

In [71]:

```

columns1 = {'stamina', 'attack_value','defense_value','capture_rate','flee_rate',
columns2 = {'attack_value','defense_value','capture_rate','flee_rate','spawn_char
columns3 = {'defense_value','capture_rate','flee_rate','spawn_chance'}
columns4 = {'capture_rate','flee_rate','spawn_chance'}
columns5 = {'flee_rate','spawn_chance'}
for i in columns2:
  plt.scatter(trainingdata.stamina,trainingdata[i])
  plt.title("Scatter plot between stamina and " +i+ "")
  plt.xlabel("stamina")
  plt.ylabel(i)
  plt.tight_layout
  plt.show()
  r,p= pearsonr(trainingdata.stamina,trainingdata[i])
  print("Pearson's correlation between stamina and " +i+ " is " + format(r))

for i in columns3:
  plt.scatter(trainingdata.attack_value,trainingdata[i])
  plt.title("Scatter plot between attack_value and " +i+ "")
  plt.xlabel("attack_value")
  plt.ylabel(i)
  plt.tight_layout
  plt.show()
  r,p= pearsonr(trainingdata.attack_value,trainingdata[i])
  print("Pearson's correlation between attack_value and " +i+ " is " + format(r)

for i in columns4:
  plt.scatter(trainingdata.defense_value,trainingdata[i])
  plt.title("Scatter plot between defense_value and " +i+ "")
  plt.xlabel("defense_value")
  plt.ylabel(i)
  plt.tight_layout
  plt.show()
  r,p= pearsonr(trainingdata.defense_value,trainingdata[i])
  print("Pearson's correlation between defense_value and " +i+ " is " + format(r

for i in columns5:
  plt.scatter(trainingdata.capture_rate,trainingdata[i])
  plt.title("Scatter plot between capture_rate and " +i+ "")
  plt.xlabel("capture_rate")
  plt.ylabel(i)
  plt.tight_layout
  plt.show()
  r,p= pearsonr(trainingdata.capture_rate,trainingdata[i])
  print("Pearson's correlation between capture_rate and " +i+ " is " + format(r)

plt.scatter(trainingdata.flee_rate,trainingdata.spawn_chance)
plt.title("Scatter plot between flee_rate and spawn_chance")
plt.xlabel("flee_rate")
plt.ylabel("spawn_chance")
plt.tight_layout
plt.show()
r,p= pearsonr(trainingdata.flee_rate,trainingdata.spawn_chance)
print("Pearson's correlation between capture_rate and spawn chance is " + format(
```



```
Pearson's correlation between stamina and attack_value is 0.3029949826738916
```

Scatter plot between stamina and flee_rate

In [72]:

```
#2.iv
onehot = []
trainingdata = trainingdata.drop('name',axis=1)
obj_trainingdata = trainingdata.select_dtypes(include=['object']).copy()
print(obj_trainingdata.value_counts())
onehot = pd.get_dummies(obj_trainingdata, columns=["primary_strength"])
onehot.head()
```

```
primary_strength
Water               28
Normal              22
Poison              14
Grass               12
Bug                 12
Fire                11
Rock                 9
Ground               8
Electric             8
Fighting             7
Psychic              6
Ghost                3
Dragon               3
Fairy                2
Ice                  1
dtype: int64
```

Out[72]:

	primary_strength_Bug	primary_strength_Dragon	primary_strength_Electric	primary_strength_Fairy
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

In [73]:

```
categoric_data = (trainingdata[categoric_colmns[0]])

colmns = list(set(list(categoric_data['primary_strength'])))

print(colmns)
for x in range(len(colmns)):
  inputdata[colmns[x]] = 0.0


for x in range(len(colmns)):
  for y in range(len(inputdata[colmns[x]])):
    if(inputdata.iloc[y][categoric_colmns[0]].values[0] == colmns[x]):
      inputdata.at[y, colmns[x]] = 1.0


print(inputdata.columns)


inputdata.head()
```

```
['Psychic', 'Dragon', 'Grass', 'Water', 'Ground', 'Ghost', 'Ice', 'Fairy', 'Bu
g', 'Electric', 'Normal', 'Fire', 'Rock', 'Fighting', 'Poison']
Index(['stamina', 'attack_value', 'defense_value', 'capture_rate', 'flee_rate',
       'spawn_chance', 'primary_strength', 'Psychic', 'Dragon', 'Grass',
       'Water', 'Ground', 'Ghost', 'Ice', 'Fairy', 'Bug', 'Electric', 'Normal',
       'Fire', 'Rock', 'Fighting', 'Poison'],
      dtype='object')
```

Out[73]:

	stamina	attack_value	defense_value	capture_rate	flee_rate	spawn_chance	primary_strength
0	90	126	126	0.16	0.10	69.0	Grass
1	120	156	158	0.08	0.07	4.2	Grass
2	160	198	200	0.04	0.05	1.7	Grass
3	78	128	108	0.16	0.10	25.3	Fire
4	116	160	140	0.08	0.07	1.2	Fire

In [74]:
```
from sklearn.utils import shuffle

y_initial = trainingdata['combat_point']

trainingdata = inputdata

trainingdata['combat_point'] = y_initial

inputdata.pop('primary_strength')
```

Out[74]:
```
0       Grass
1       Grass
2       Grass
3        Fire
4        Fire
        ...  
141      Rock
142    Normal
143    Dragon
144    Dragon
145    Dragon
Name: primary_strength, Length: 146, dtype: object
```

In [75]:
```
trainingdata.insert(0,'bias', 1)
trainingdata
print(trainingdata.columns)
```

```
Index(['bias', 'stamina', 'attack_value', 'defense_value', 'capture_rate',
       'flee_rate', 'spawn_chance', 'Psychic', 'Dragon', 'Grass', 'Water',
       'Ground', 'Ghost', 'Ice', 'Fairy', 'Bug', 'Electric', 'Normal', 'Fire',
       'Rock', 'Fighting', 'Poison', 'combat_point'],
      dtype='object')
```

In [76]:

```
#implementationoflinearregression
from sklearn.utils import shuffle

def OLS(train_x,train_y, l):            #OLS gives the ordinary least square solu
  train_x = train_x.to_numpy()
  train_y = train_y.to_numpy()
  train_x_transpose = np.transpose(train_x)
  x_val = np.matmul(train_x_transpose, train_x)

  identity_matrix = np.identity(x_val.shape[0],dtype=int)
  identity_matrix = identity_matrix*l
  x_val = np.add(x_val,identity_matrix)
  x_inverse = np.linalg.pinv(x_val)
  x = np.matmul(train_x_transpose, train_y)

  w = np.matmul(x_inverse, x)

  w = np.matmul(x_inverse, x)
  return w



def valueRss(test_x, w, test_y):   #RSS value is calculated
  test_x = test_x.to_numpy()
  test_y = test_y.to_numpy()
  w1 = np.transpose(w)
  test_x = np.transpose(test_x)
  pred_y = np.matmul(w1,test_x);
  #print(y_pred)
  test_y = np.transpose(test_y)

  rssvalue = np.sqrt(np.sum(np.square(test_y-pred_y)))

  return rssvalue



def linearregression(trdata,l,parts=5): #since it is asked to divide into 5 parts
  rss = 0
  for i in range(0,parts):
    trdata = shuffle(trdata)
    s = int(len(trdata)/5)
    test_data = trdata[:s]    #1/5th trainingdata is assigned to testdata
    train_data = trdata[s:]   #4/5th trainingdata is assigned as trainingdata
    train_x = train_data.loc[:,:'Normal']
    test_x = test_data.loc[:,:'Normal']
    train_y = train_data.loc[:,'combat_point':]
    test_y = test_data.loc[:,'combat_point':]
    w = OLS(train_x,train_y, l)
    rssfolds = valueRss(test_x, w, test_y)
    print('The value of Square root of RSS for the', i+1, 'fold is ', rssfolds)
    rss += rssfolds
  print('Average Square root of RSS over all folds is', rss/5)

linearregression(trainingdata,0)
```

```
The value of Square root of RSS for the 1 fold is  633.3252599289458
The value of Square root of RSS for the 2 fold is  813.1303274096778
The value of Square root of RSS for the 3 fold is  621.8472141492895
The value of Square root of RSS for the 4 fold is  531.1145578319313
The value of Square root of RSS for the 5 fold is  813.1519918468102
Average Square root of RSS over all folds is 682.5138702333309
```

In [77]:

```
#vi
print("For different lambda values:")
lam = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]
for i in lam :
  linearregression(trainingdata,math.exp(-i))
```

```
For different lambda values:
The value of Square root of RSS for the 1 fold is  682.0322568136514
The value of Square root of RSS for the 2 fold is  759.8741526871486
The value of Square root of RSS for the 3 fold is  871.4049567210126
The value of Square root of RSS for the 4 fold is  700.1001168429558
The value of Square root of RSS for the 5 fold is  2129.835558806512
Average Square root of RSS over all folds is 1028.649408374256
The value of Square root of RSS for the 1 fold is  458.1978883837772
The value of Square root of RSS for the 2 fold is  808.811249384208
The value of Square root of RSS for the 3 fold is  640.1892847103851
The value of Square root of RSS for the 4 fold is  573.3563012070335
The value of Square root of RSS for the 5 fold is  748.980807195927
Average Square root of RSS over all folds is 645.9071061762662
The value of Square root of RSS for the 1 fold is  804.8730767563574
The value of Square root of RSS for the 2 fold is  740.0863729819672
The value of Square root of RSS for the 3 fold is  660.0931906025138
The value of Square root of RSS for the 4 fold is  1685.5512711617855
The value of Square root of RSS for the 5 fold is  786.347426470503
Average Square root of RSS over all folds is 935.3902675946254
The value of Square root of RSS for the 1 fold is  599.9992521978578
The value of Square root of RSS for the 2 fold is  723.1028485614488
The value of Square root of RSS for the 3 fold is  873.5269856700314
The value of Square root of RSS for the 4 fold is  591.7817111815367
The value of Square root of RSS for the 5 fold is  796.5724575878228
Average Square root of RSS over all folds is 716.9966510397395
The value of Square root of RSS for the 1 fold is  871.0083385009282
The value of Square root of RSS for the 2 fold is  693.0348121904799
The value of Square root of RSS for the 3 fold is  607.9622343328747
The value of Square root of RSS for the 4 fold is  777.6659803280144
The value of Square root of RSS for the 5 fold is  1908.789234985276
Average Square root of RSS over all folds is 971.6921200675148
The value of Square root of RSS for the 1 fold is  1632.2281966542566
The value of Square root of RSS for the 2 fold is  820.846819812046
The value of Square root of RSS for the 3 fold is  491.67810477799384
The value of Square root of RSS for the 4 fold is  902.651354580054
The value of Square root of RSS for the 5 fold is  636.3972250374236
Average Square root of RSS over all folds is 896.7603401723547
The value of Square root of RSS for the 1 fold is  1625.081652902264
The value of Square root of RSS for the 2 fold is  1572.601314058802
The value of Square root of RSS for the 3 fold is  659.7657206879231
The value of Square root of RSS for the 4 fold is  662.2597748304718
The value of Square root of RSS for the 5 fold is  747.6562353681686
Average Square root of RSS over all folds is 1053.4729395695258
The value of Square root of RSS for the 1 fold is  580.4815113601517
The value of Square root of RSS for the 2 fold is  674.34550524647
The value of Square root of RSS for the 3 fold is  805.1621724062684
The value of Square root of RSS for the 4 fold is  607.5838033877625
The value of Square root of RSS for the 5 fold is  1618.7143635257312
Average Square root of RSS over all folds is 857.2574711852769
The value of Square root of RSS for the 1 fold is  625.0435512392652
The value of Square root of RSS for the 2 fold is  606.9331379146496
The value of Square root of RSS for the 3 fold is  625.6382675112036
The value of Square root of RSS for the 4 fold is  1571.3187590500816
The value of Square root of RSS for the 5 fold is  609.3391649011256
Average Square root of RSS over all folds is 807.6545761232651
The value of Square root of RSS for the 1 fold is  604.8701752122826
The value of Square root of RSS for the 2 fold is  1632.6778824151309
The value of Square root of RSS for the 3 fold is  631.3155293638558
The value of Square root of RSS for the 4 fold is  639.189951219601
The value of Square root of RSS for the 5 fold is  826.9462467990833
Average Square root of RSS over all folds is 866.9999570019907
```

In [121]:

```python
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

samplemean = np.mean(trainingdata['combat_point'])

y_val = list(trainingdata['combat_point'])

y_data = []
for i in y_val:
  if(i < int(samplemean)):
    y_data.append(0)
  else:
    y_data.append(1)

y_data = pd.DataFrame(y_data)

train_x, test_x, train_y,test_y = train_test_split(trainingdata.loc[:,:'Normal'],
clf = LogisticRegression(random_state=0,penalty='none').fit(train_x,train_y)

print(clf.score(test_x,test_y))
```

```
0.9333333333333333
```

In [115]:

```
lamda =[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]

idealparameter = 0

highestaccuracy = 0.0

Accuracyvalues = []

for c in lamda:
  acc_part = []
  for i in range(5):
    x_fold_train, x_fold_test, y_fold_train, y_fold_test = train_test_split(trai
    log_reg_r = LogisticRegression(random_state=0,penalty='l2',C=c).fit(x_fold_tr
    acc_part.append(log_reg_r.score(x_fold_test,y_fold_test))
  acc = sum(acc_part)/len(acc_part)
  Accuracyvalues.append(acc)
  if(highestaccuracy < acc):
    highestaccuracy = acc
    idealparameter = c

print(Accuracyvalues)

print('Ideal Hyper paramenter obtained for value ', idealparameter, ', Accuracy
```

```
[0.9583333333333334, 0.95, 0.975, 0.9916666666666666, 0.9333333333333332, 0.966
6666666666668, 0.975, 0.9666666666666666, 0.9833333333333334, 0.983333333333333
4]
Ideal Hyper paramenter obtained for value  0.4 , Accuracy obtained for this hyp
er parameter is equal to,  0.9916666666666666
```