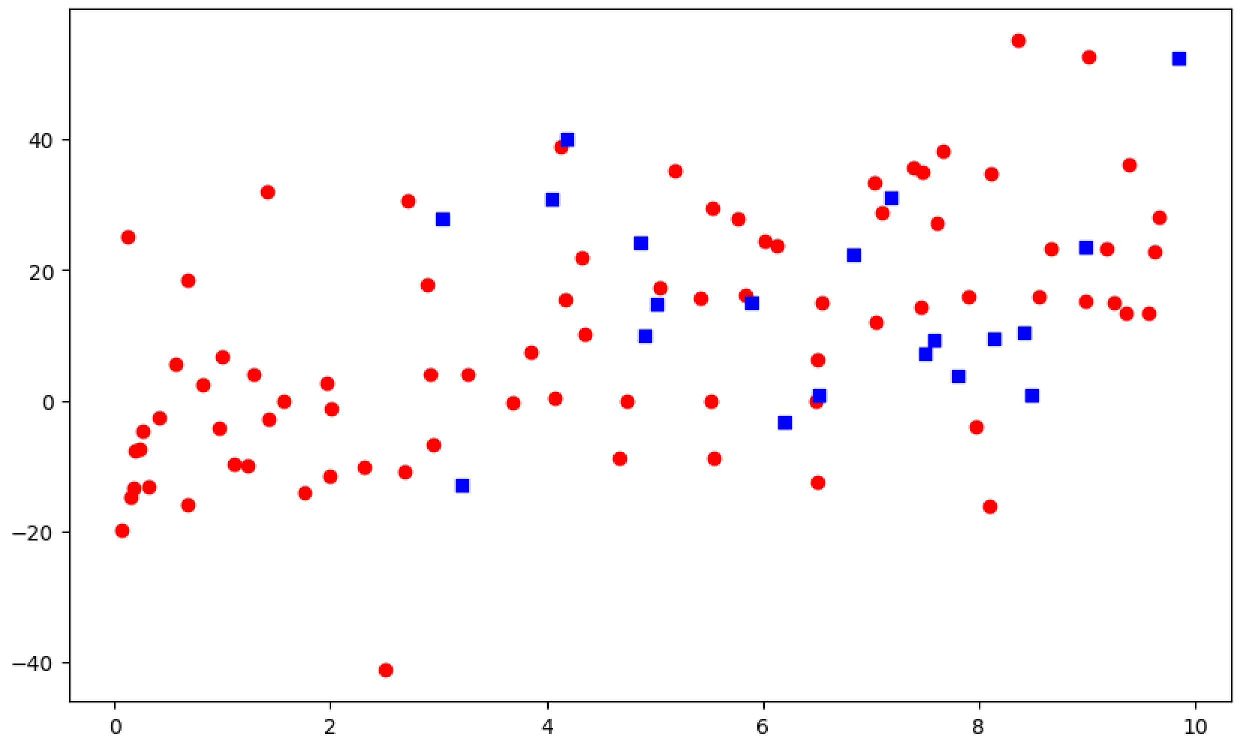
- Sahan Abeyrathna
- 210005H
- Assignment 1
- ENTC

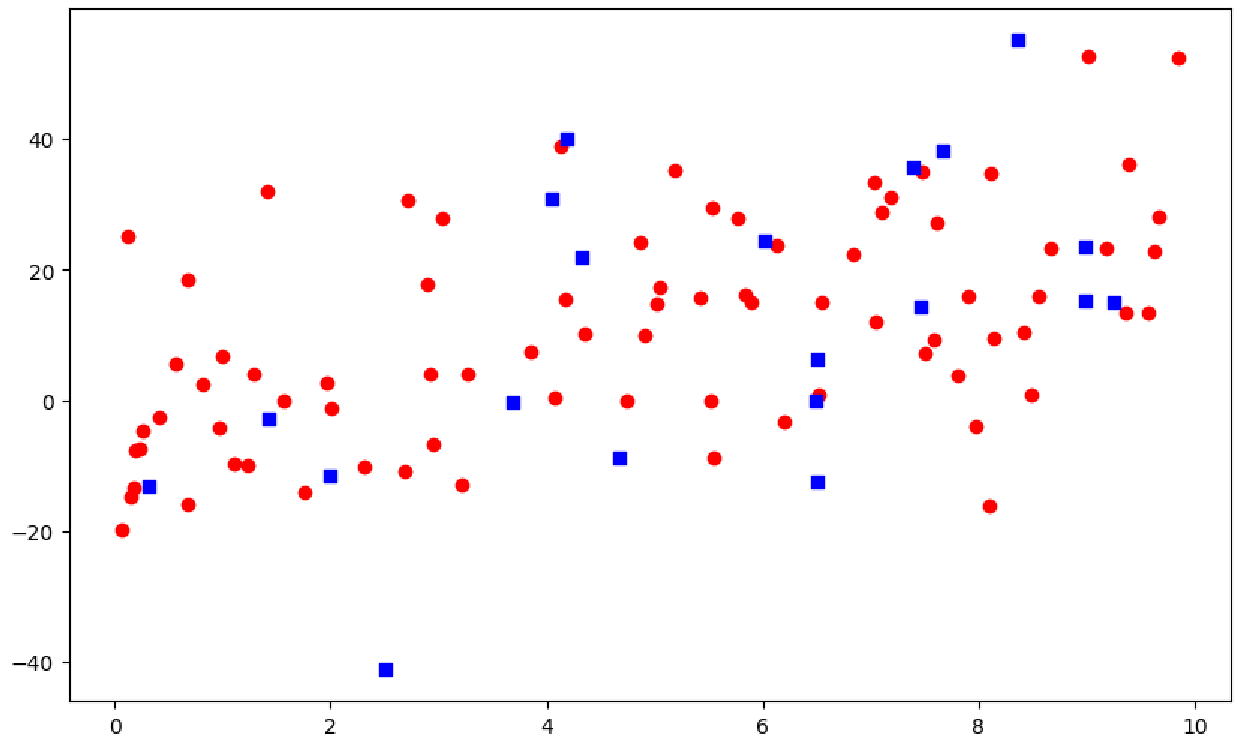
Learning from data and related challenges and linear models for regression

Learning from data

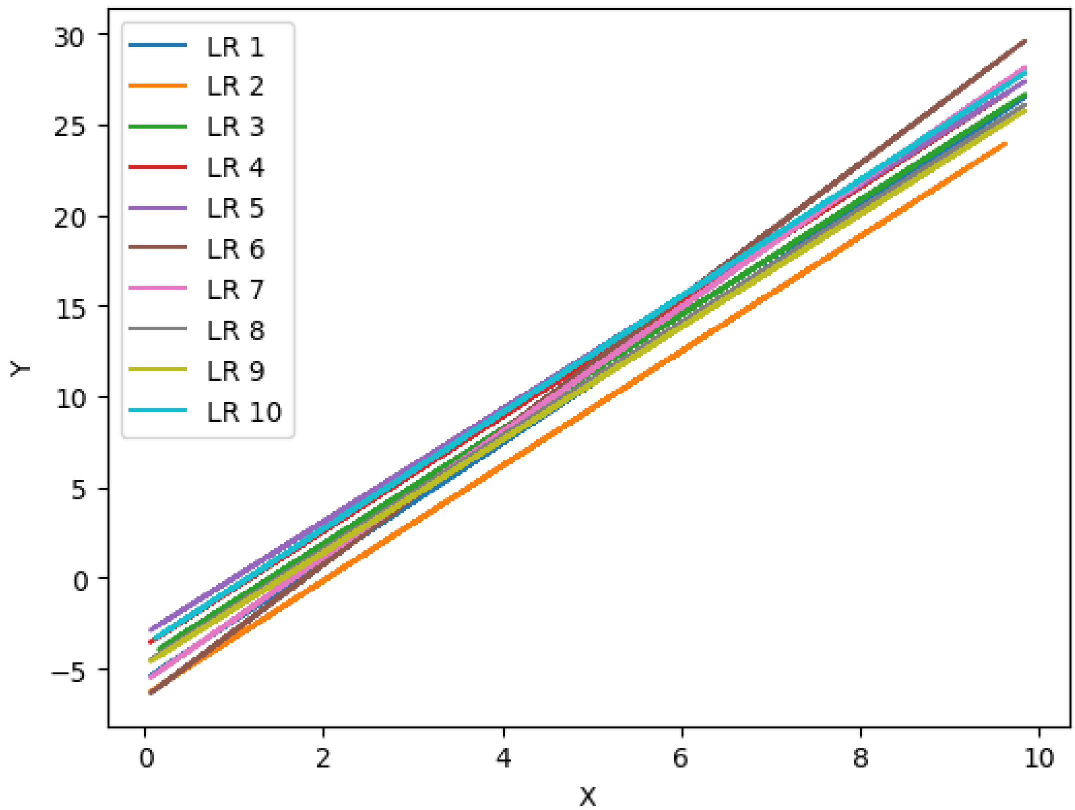
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
# Generate 100 samples
n_samples = 100
# Generate X values (uniformly distributed between 0 and 10)
X = 10 * np.random.rand(n_samples, 1)
# Generate epsilon values (normally distributed with mean 0 and
standard deviation 15)
epsilon = np.random.normal(0, 15, n_samples)
# Generate Y values using the model Y = 3 + 3X + epsilon
Y = 3 + 2 * X + epsilon[:, np.newaxis]

r=np.random.randint(104)
# Split the data into training and test sets (80% train,20% test)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=r)
# Plot the data points
plt.figure(figsize=(10, 6))
plt.scatter(X_train, Y_train, alpha=1,
marker='o',color='red',label='Training Data')
plt.scatter(X_test, Y_test, alpha=1,
marker='s',color='blue',label='Testing Data')
plt.show()
```




```
r=np.random.randint(104)
# Split the data into training and test sets (80% train,20% test)
X_train, X_test, Y_train, Y_test = train_test_split(X,
Y,test_size=0.2, random_state=r)
# Plot the data points
plt.figure(figsize=(10, 6))
plt.scatter(X_train, Y_train, alpha=1,
marker='o',color='red',label='Training Data')
plt.scatter(X_test, Y_test, alpha=1,
marker='s',color='blue',label='Testing Data')
plt.show()
```

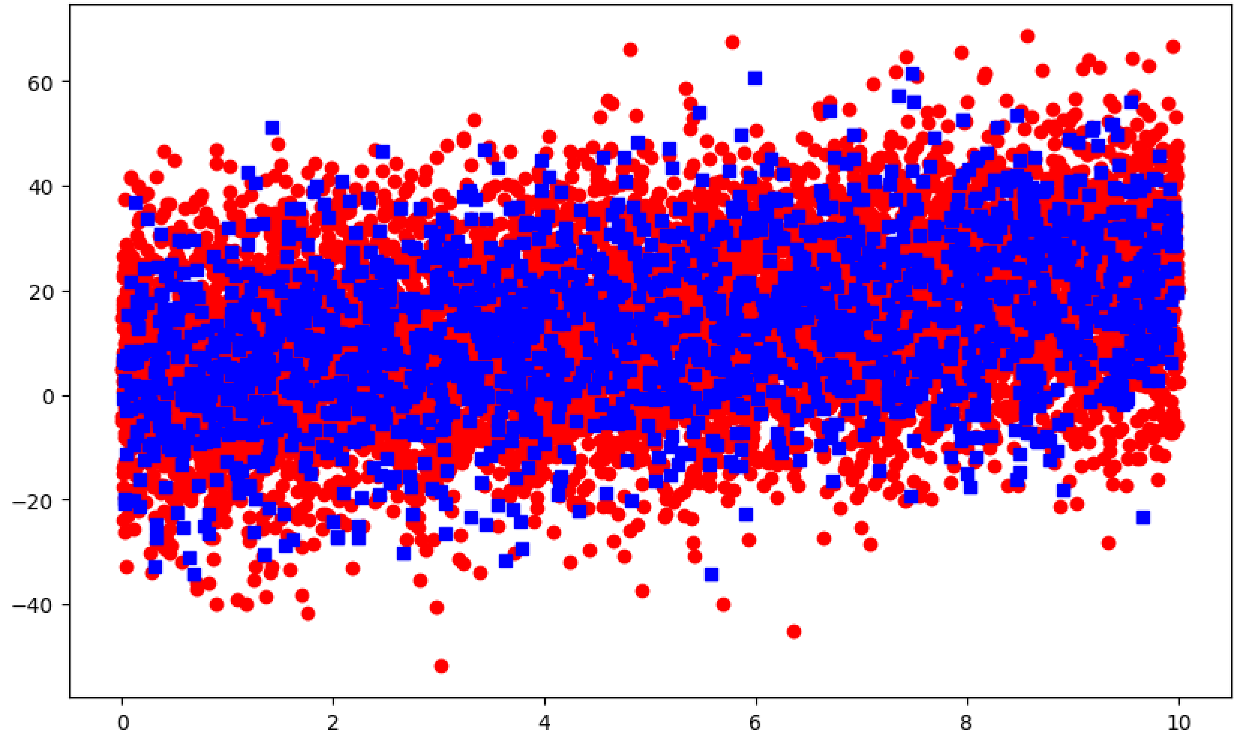



```
for i in range(10): # Plotting 10 different instances
  X_train, X_test, Y_train, Y_test = train_test_split(X,Y,
test_size=0.2, random_state=np.random.randint(104))
  model = LinearRegression()
  model.fit(X_train, Y_train)
  Y_pred_train = model.predict(X_train)
  plt.plot(X_train, Y_pred_train, label=f'LR {i+1}')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()
```

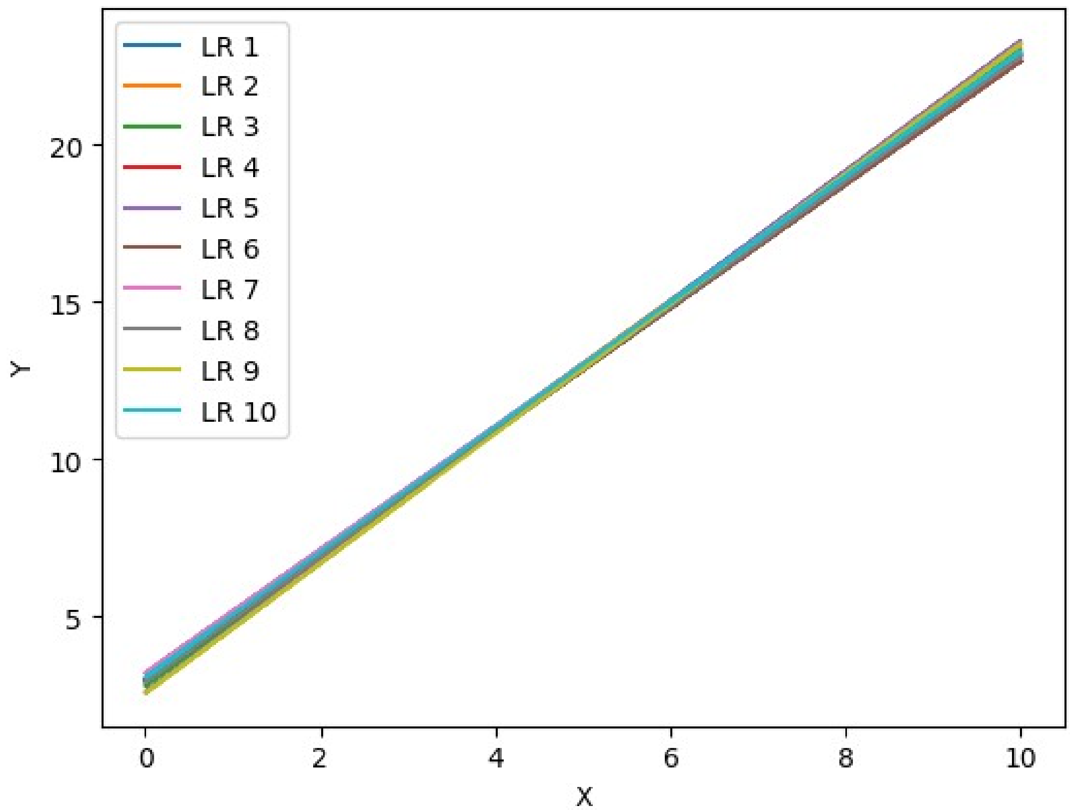



```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
# Generate 100 samples
n_samples = 10000
# Generate X values (uniformly distributed between 0 and 10)
X = 10 * np.random.rand(n_samples, 1)
# Generate epsilon values (normally distributed with mean 0 and
standard deviation 15)
epsilon = np.random.normal(0, 15, n_samples)
# Generate Y values using the model Y = 3 + 3X + epsilon
Y = 3 + 2 * X + epsilon[:, np.newaxis]

r=np.random.randint(104)
# Split the data into training and test sets (80% train,20% test)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.2, random_state=r)
# Plot the data points
plt.figure(figsize=(10, 6))
plt.scatter(X_train, Y_train, alpha=1,
marker='o',color='red',label='Training Data')
plt.scatter(X_test, Y_test, alpha=1,
```

```
marker='s',color='blue',label='Testing Data')
plt.show()
```




```
for i in range(10): # Plotting 10 different instances
  X_train, X_test, Y_train, Y_test = train_test_split(X,Y,
test_size=0.2, random_state=np.random.randint(104))
  model = LinearRegression()
  model.fit(X_train, Y_train)
  Y_pred_train = model.predict(X_train)
  plt.plot(X_train, Y_pred_train, label=f'LR {i+1}')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()
```




```
pip install ucimlrepo
```

```
Collecting ucimlrepo
  Downloading ucimlrepo-0.0.7-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: pandas>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from ucimlrepo) (2.1.4)
Requirement already satisfied: certifi>=2020.12.5 in
/usr/local/lib/python3.10/dist-packages (from ucimlrepo) (2024.8.30)
Requirement already satisfied: numpy<2,>=1.22.4 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0-
>ucimlrepo) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0-
>ucimlrepo) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0-
>ucimlrepo) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in
/usr/local/lib/python3.10/dist-packages (from pandas>=1.0.0-
>ucimlrepo) (2024.1)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2-
>pandas>=1.0.0->ucimlrepo) (1.16.0)
Downloading ucimlrepo-0.0.7-py3-none-any.whl (8.0 kB)
```

```
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.7
```

```python
from ucimlrepo import fetch_ucirepo
# fetch dataset
infrared_thermography_temperature = fetch_ucirepo(id=925)
# data (as pandas dataframes)
X = infrared_thermography_temperature.data.features
y = infrared_thermography_temperature.data.targets
# metadata
print(infrared_thermography_temperature.metadata)
# variable information
print(infrared_thermography_temperature.variables)
```


```
{'uci_id': 925, 'name': 'Infrared Thermography Temperature',
'repository_url':
'https://archive.ics.uci.edu/dataset/925/infrared+thermography+tempera
ture+dataset', 'data_url':
'https://archive.ics.uci.edu/static/public/925/data.csv', 'abstract':
'The Infrared Thermography Temperature Dataset contains temperatures
read from various locations of inferred images about patients, with
the addition of oral temperatures measured for each individual. The 33
features consist of gender, age, ethnicity, ambiant temperature,
humidity, distance, and other temperature readings from the thermal
images. The dataset is intended to be used in a regression task to
predict the oral temperature using the environment information as well
as the thermal image readings. ', 'area': 'Health and Medicine',
'tasks': ['Regression'], 'characteristics': ['Tabular'],
'num_instances': 1020, 'num_features': 33, 'feature_types': ['Real',
'Categorical'], 'demographics': ['Gender', 'Age', 'Ethnicity'],
'target_col': ['aveOralF', 'aveOralM'], 'index_col': ['SubjectID'],
'has_missing_values': 'no', 'missing_values_symbol': None,
'year_of_dataset_creation': 2021, 'last_updated': 'Tue Dec 12 2023',
'dataset_doi': '10.13026/9ay4-2c37', 'creators': ['Quanzeng Wang',
'Yangling Zhou', 'Pejman Ghassemi', 'David McBride', 'J. Casamento',
'T. Pfefer', 'Quanzeng Wang', 'Yangling Zhou', 'Pejman Ghassemi',
'David McBride', 'J. Casamento', 'T. Pfefer'], 'intro_paper':
{'title': 'Infrared Thermography for Measuring Elevated Body
Temperature: Clinical Accuracy, Calibration, and Evaluation',
'authors': 'Quanzeng Wang, Yangling Zhou, Pejman Ghassemi, David
McBride, J. Casamento, T. Pfefer', 'published_in': 'Italian National
Conference on Sensors', 'year': 2021, 'url':
'https://www.semanticscholar.org/paper/443b9932d295ca3a014e7d874b4bd77
a33a276bd', 'doi': None}, 'additional_info': {'summary': None,
'purpose': None, 'funded_by': None, 'instances_represent': None,
'recommended_data_splits': None, 'sensitive_data': None,
'preprocessing_description': None, 'variable_info': '- gender\n- age\
n- ethnicity\n- ambiant temperature\n- humidity\n- distance\n-
temperature readings from the thermal images', 'citation': None},
'external_url':
```

```
'https://physionet.org/content/face-oral-temp-data/1.0.0/'}
            name     role         type demographic  \
0      SubjectID       ID  Categorical        None
1       aveOralF   Target   Continuous        None
2       aveOralM   Target   Continuous        None
3         Gender  Feature  Categorical      Gender
4            Age  Feature  Categorical         Age
5      Ethnicity  Feature  Categorical   Ethnicity
6          T_atm  Feature   Continuous        None
7       Humidity  Feature   Continuous        None
8       Distance  Feature   Continuous        None
9      T_offset1  Feature   Continuous        None
10     Max1R13_1  Feature   Continuous        None
11     Max1L13_1  Feature   Continuous        None
12   aveAllR13_1  Feature   Continuous        None
13   aveAllL13_1  Feature   Continuous        None
14         T_RC1  Feature   Continuous        None
15     T_RC_Dry1  Feature   Continuous        None
16     T_RC_Wet1  Feature   Continuous        None
17     T_RC_Max1  Feature   Continuous        None
18         T_LC1  Feature   Continuous        None
19     T_LC_Dry1  Feature   Continuous        None
20     T_LC_Wet1  Feature   Continuous        None
21     T_LC_Max1  Feature   Continuous        None
22          RCC1  Feature   Continuous        None
23          LCC1  Feature   Continuous        None
24    canthiMax1  Feature   Continuous        None
25   canthi4Max1  Feature   Continuous        None
26        T_FHCC1  Feature   Continuous        None
27        T_FHRC1  Feature   Continuous        None
28        T_FHLC1  Feature   Continuous        None
29        T_FHBC1  Feature   Continuous        None
30        T_FHTC1  Feature   Continuous        None
31     T_FH_Max1  Feature   Continuous        None
32    T_FHC_Max1  Feature   Continuous        None
33        T_Max1  Feature   Continuous        None
34         T_OR1  Feature   Continuous        None
35     T_OR_Max1  Feature   Continuous        None

                                          description units
missing_values
0                                          Subject ID  None
no
1                Oral temperature measured in fast mode  None
no
2             Oral temperature measured in monitor mode  None
no
3                                      Male or Female  None
no
```

```
4                             Age ranges in categories\n  None
no
5   American Indian or Alaska Native, Asian, Black...  None
no
6                                  Ambiant temperature  None
no
7                                    Relative humidity  None
no
8        Distance between the subjects and the IRTs.    None
no
9   Temperature difference between the set and mea...  None
no
10  Max value of a circle with diameter of 13 pixe...  None
no
11  Max value of a circle with diameter of 13 pixe...  None
no
12  Average value of a circle with diameter of 13 ...  None
no
13  Average value of a circle with diameter of 13 ...  None
no
14  Average temperature of the highest four pixels...  None
no
15  Average temperature of the highest four pixels...  None
no
16  Average temperature of the highest four pixels...  None
no
17  Max value of a square of 24x24 pixels around t...  None
no
18  Average temperature of the highest four pixels...  None
no
19  Average temperature of the highest four pixels...  None
no
20  Average temperature of the highest four pixels...  None
no
21  Max value of a circle with diameter of 13 pixe...  None
no
22  Average value of a square of 3x3 pixels center...  None
no
23  Average value of a square of 3x3 pixels center...  None
no
24                  Max value in the extended canthi area  None
no
25  Average temperature of the highest four pixels...  None
no
26  Average value in the center point of forehead,...  None
no
27  Average value in the right point of the forehe...  None
no
28  Average value in the left point of the forehea...  None
```

```
no
29  Average value in the bottom point of the foreh...  None
no
30  Average value in the top point of the forehead...  None
no
31  Maximum temperature within the extended forehe...  None
no
32  Max value in the center point of forehead, a s...  None
no
33  Maximum temperature within the whole face region.  None
no
34  Average temperature of the highest four pixels...  None
no
35       Maximum temperature within the mouth region.  None
no
```

```
print(infrared_thermography_temperature.data)
```

```
{'ids':       SubjectID
0      161117-1
1      161117-2
2      161117-3
3      161117-4
4      161117-5
...         ...
1015  180425-05
1016  180425-06
1017  180502-01
1018  180507-01
1019  180514-01

[1020 rows x 1 columns], 'features':       Gender    Age
Ethnicity  T_atm  Humidity  Distance  \
0       Male  41-50                      White   24.0      28.0
0.8
1     Female  31-40  Black or African-American   24.0      26.0
0.8
2     Female  21-30                      White   24.0      26.0
0.8
3     Female  21-30  Black or African-American   24.0      27.0
0.8
4       Male  18-20                      White   24.0      27.0
0.8
...      ...    ...                        ...    ...       ...
...
1015  Female  21-25                      Asian   25.7      50.8
0.6
1016  Female  21-25                      White   25.7      50.8
0.6
1017  Female  18-20  Black or African-American   28.0      24.3
```

```
0.6
1018    Male  26-30            Hispanic/Latino   25.0      39.8
0.6
1019  Female  18-20                      White   23.8      45.6
0.6

      T_offset1  Max1R13_1  Max1L13_1  aveAllR13_1  ...  T_FHCC1
T_FHRC1  \
0        0.7025    35.0300    35.3775      34.4000  ...  33.5775
33.4775
1        0.7800    34.5500    34.5200      33.9300  ...  34.0325
34.0550
2        0.8625    35.6525    35.5175      34.2775  ...  34.9000
34.8275
3        0.9300    35.2225    35.6125      34.3850  ...  34.4400
34.4225
4        0.8950    35.5450    35.6650      34.9100  ...  35.0900
35.1600
...         ...        ...        ...          ...  ...      ...
...
1015     1.2225    35.6425    35.6525      34.8575  ...  35.1075
35.3475
1016     1.4675    35.9825    35.7575      35.4275  ...  35.3100
35.2175
1017     0.1300    36.4075    36.3400      35.8700  ...  35.4350
35.2400
1018     1.2450    35.8150    35.5250      34.2950  ...  34.8400
35.0200
1019     0.8675    35.7075    35.5825      34.8875  ...  34.5475
34.6500

      T_FHLC1  T_FHBC1  T_FHTC1  T_FH_Max1  T_FHC_Max1   T_Max1
T_OR1  \
0     33.3725  33.4925  33.0025    34.5300     34.0075  35.6925
35.6350
1     33.6775  33.9700  34.0025    34.6825     34.6600  35.1750
35.0925
2     34.6475  34.8200  34.6700    35.3450     35.2225  35.9125
35.8600
3     34.6550  34.3025  34.9175    35.6025     35.3150  35.7200
34.9650
4     34.3975  34.6700  33.8275    35.4175     35.3725  35.8950
35.5875
...       ...      ...      ...        ...         ...      ...      .
..
1015  35.4000  35.1375  35.2750    35.8525     35.7475  36.0675
35.6775
1016  35.2200  35.2075  35.0700    35.7650     35.5525  36.5000
36.4525
```

```
1017  35.2275  35.3675  35.3425    36.3750     35.7100  36.5350
35.9650
1018  34.9250  34.7150  34.5950    35.4150     35.3100  35.8600
35.4150
1019  34.6700  34.2150  34.7100    35.1525     35.1175  35.9725
35.8900

      T_OR_Max1
0       35.6525
1       35.1075
2       35.8850
3       34.9825
4       35.6175
...         ...
1015    35.7100
1016    36.4900
1017    35.9975
1018    35.4350
1019    35.9175

[1020 rows x 33 columns], 'targets':       aveOralF  aveOralM
0        36.85     36.59
1        37.00     37.19
2        37.20     37.34
3        36.85     37.09
4        36.80     37.04
...        ...       ...
1015     36.95     36.99
1016     37.25     37.19
1017     37.35     37.59
1018     37.15     37.29
1019     37.05     37.19

[1020 rows x 2 columns], 'original':       SubjectID  aveOralF
aveOralM  Gender    Age                    Ethnicity  \
0      161117-1     36.85     36.59    Male  41-50
White
1      161117-2     37.00     37.19  Female  31-40  Black or African-
American
2      161117-3     37.20     37.34  Female  21-30
White
3      161117-4     36.85     37.09  Female  21-30  Black or African-
American
4      161117-5     36.80     37.04    Male  18-20
White
...         ...       ...       ...     ...    ...
...
1015  180425-05     36.95     36.99  Female  21-25
Asian
1016  180425-06     37.25     37.19  Female  21-25
```

```
White
1017  180502-01     37.35     37.59  Female  18-20  Black or African-
American
1018  180507-01     37.15     37.29    Male  26-30
Hispanic/Latino
1019  180514-01     37.05     37.19  Female  18-20
White

      T_atm  Humidity  Distance  T_offset1  ...  T_FHCC1  T_FHRC1
T_FHLC1  \
0      24.0      28.0       0.8     0.7025  ...  33.5775  33.4775
33.3725
1      24.0      26.0       0.8     0.7800  ...  34.0325  34.0550
33.6775
2      24.0      26.0       0.8     0.8625  ...  34.9000  34.8275
34.6475
3      24.0      27.0       0.8     0.9300  ...  34.4400  34.4225
34.6550
4      24.0      27.0       0.8     0.8950  ...  35.0900  35.1600
34.3975
...     ...       ...       ...        ...  ...      ...      ...
...
1015   25.7      50.8       0.6     1.2225  ...  35.1075  35.3475
35.4000
1016   25.7      50.8       0.6     1.4675  ...  35.3100  35.2175
35.2200
1017   28.0      24.3       0.6     0.1300  ...  35.4350  35.2400
35.2275
1018   25.0      39.8       0.6     1.2450  ...  34.8400  35.0200
34.9250
1019   23.8      45.6       0.6     0.8675  ...  34.5475  34.6500
34.6700

      T_FHBC1  T_FHTC1  T_FH_Max1  T_FHC_Max1   T_Max1    T_OR1
T_OR_Max1
0     33.4925  33.0025    34.5300     34.0075  35.6925  35.6350
35.6525
1     33.9700  34.0025    34.6825     34.6600  35.1750  35.0925
35.1075
2     34.8200  34.6700    35.3450     35.2225  35.9125  35.8600
35.8850
3     34.3025  34.9175    35.6025     35.3150  35.7200  34.9650
34.9825
4     34.6700  33.8275    35.4175     35.3725  35.8950  35.5875
35.6175
...       ...      ...        ...         ...      ...      ...
...
1015  35.1375  35.2750    35.8525     35.7475  36.0675  35.6775
35.7100
```

```
1016  35.2075  35.0700    35.7650     35.5525  36.5000  36.4525
36.4900
1017  35.3675  35.3425    36.3750     35.7100  36.5350  35.9650
35.9975
1018  34.7150  34.5950    35.4150     35.3100  35.8600  35.4150
35.4350
1019  34.2150  34.7100    35.1525     35.1175  35.9725  35.8900
35.9175

[1020 rows x 36 columns], 'headers': Index(['SubjectID', 'aveOralF',
'aveOralM', 'Gender', 'Age', 'Ethnicity',
       'T_atm', 'Humidity', 'Distance', 'T_offset1', 'Max1R13_1',
'Max1L13_1',
       'aveAllR13_1', 'aveAllL13_1', 'T_RC1', 'T_RC_Dry1',
'T_RC_Wet1',
       'T_RC_Max1', 'T_LC1', 'T_LC_Dry1', 'T_LC_Wet1', 'T_LC_Max1',
'RCC1',
       'LCC1', 'canthiMax1', 'canthi4Max1', 'T_FHCC1', 'T_FHRC1',
'T_FHLC1',
       'T_FHBC1', 'T_FHTC1', 'T_FH_Max1', 'T_FHC_Max1', 'T_Max1',
'T_OR1',
       'T_OR_Max1'],
      dtype='object')}
```

```
import pandas as pd
print(f"X shape beforeremoval: {X.shape}")
print(f"y shape beforeremoval: {y.shape}")
df =pd.concat([X,y],axis =1)
df =df.dropna()
X =df.iloc[:, :-2]
y =df.iloc[:,-2:]
print(f"X shape afterremoval:{X.shape}")
print(f"y shape afterremoval:{y.shape}")
```

```
X shape beforeremoval: (1020, 33)
y shape beforeremoval: (1020, 2)
X shape afterremoval:(1018, 33)
y shape afterremoval:(1018, 2)
```

```
target =y['aveOralM']
features= X[['Age', 'T_atm','Humidity','T_Max1','T_offset1']]
print(features)
```

```
        Age  T_atm  Humidity   T_Max1  T_offset1
0     41-50   24.0      28.0  35.6925     0.7025
1     31-40   24.0      26.0  35.1750     0.7800
2     21-30   24.0      26.0  35.9125     0.8625
3     21-30   24.0      27.0  35.7200     0.9300
4     18-20   24.0      27.0  35.8950     0.8950
...     ...    ...       ...      ...        ...
```

```
1015  21-25   25.7      50.8  36.0675     1.2225
1016  21-25   25.7      50.8  36.5000     1.4675
1017  18-20   28.0      24.3  36.5350     0.1300
1018  26-30   25.0      39.8  35.8600     1.2450
1019  18-20   23.8      45.6  35.9725     0.8675

[1018 rows x 5 columns]
```

```
def convert_age_range(age_range):
  """Converts the age rangetoasingleaveragevalue"""
  if '>' in age_range:
    return int(age_range.replace('>','').strip())
  lower,upper =map(int,age_range.split('-'))
  return np.mean([lower,upper])

features['Age']= features['Age'].apply(convert_age_range)
print(features)
```

```
       Age  T_atm  Humidity   T_Max1  T_offset1
0     45.5   24.0      28.0  35.6925     0.7025
1     35.5   24.0      26.0  35.1750     0.7800
2     25.5   24.0      26.0  35.9125     0.8625
3     25.5   24.0      27.0  35.7200     0.9300
4     19.0   24.0      27.0  35.8950     0.8950
...    ...    ...       ...      ...        ...
1015  23.0   25.7      50.8  36.0675     1.2225
1016  23.0   25.7      50.8  36.5000     1.4675
1017  19.0   28.0      24.3  36.5350     0.1300
1018  28.0   25.0      39.8  35.8600     1.2450
1019  19.0   23.8      45.6  35.9725     0.8675

[1018 rows x 5 columns]

<ipython-input-23-1a38761dabf6>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  features['Age']= features['Age'].apply(convert_age_range)
```

```
X_train,X_test, y_train,y_test=
train_test_split(features,target,test_size =
0.2,random_state=np.random.randint(104))

linear_regression =LinearRegression()
linear_regression.fit(X_train,y_train)
print('Intercept:\n',linear_regression.intercept_)
print('Coefficients: \n',linear_regression.coef_)
```

```
Intercept:
 5.350271904448654
Coefficients:
 [ 0.00106708 -0.06026008  0.00099853  0.91390548  0.09962315]
```

```
features= X[['T_OR1','T_OR_Max1', 'T_FHC_Max1', 'T_FH_Max1']]
X_train,X_test, y_train,y_test=
train_test_split(features,target,test_size =
0.2,random_state=np.random.randint(104))
linear_regression =LinearRegression()
linear_regression.fit(X_train,y_train)
print('Intercept:\n',linear_regression.intercept_)
print('Coefficients: \n',linear_regression.coef_)
```

```
Intercept:
 7.1436575396059006
Coefficients:
 [-0.05575003  0.60320041 -0.05115736  0.34035037]
```

```
print(X_test.shape)
```

```
(204, 4)
```

```
import statsmodels.api as sm

residual_sum_of_squares = np.sum((linear_regression.predict(X_test)-
y_test)** 2)
residual_standard_error = np.sqrt(residual_sum_of_squares /
(X_test.shape[0]-X_test.shape[1]- 1))

mean_squared_error = np.mean((linear_regression.predict(X_test)-
y_test) ** 2)

r_squared = linear_regression.score(X_test, y_test)
X_test_with_intercept = np.c_[np.ones(X_test.shape[0]), X_test] # Add
intercept term if not included
XtX_inv = np.linalg.inv(X_test_with_intercept.T @
X_test_with_intercept)
standard_error_coefficients = np.sqrt(np.diag(residual_standard_error
** 2 * XtX_inv))

ols_model = sm.OLS(y_test, X_test_with_intercept).fit()
summary = ols_model.summary()

t_values = ols_model.tvalues.values
p_values = ols_model.pvalues.values

print(f"Residual Sum of Squares: {residual_sum_of_squares}\n")
print(f"Residual Standard Error: {residual_standard_error}\n")
print(f"Mean Squared Error: {mean_squared_error}\n")
print(f"R-Squared: {r_squared}\n")
```

```
print(f"Standard Error of Coefficients:\
n{standard_error_coefficients}\n")
print(f"t-values:\n{t_values}\n")
print(f"p-values:\n{p_values}\n")
print(f"\n\n\n\n{summary}")
```

```
Residual Sum of Squares: 20.30989897585181

Residual Standard Error: 0.31946798563940887

Mean Squared Error: 0.09955832831299906

R-Squared: 0.6428739418600805

Standard Error of Coefficients:
[1.56946255 1.80438807 1.79706711 0.08663473 0.09491936]

t-values:
[ 5.51634926  0.76840282 -0.46550164 -1.15883401  3.70343768]

p-values:
[1.06823189e-07 4.43158815e-01 6.42081296e-01 2.47912556e-01
 2.75451148e-04]




                            OLS Regression Results

======================================================================
========
Dep. Variable:               aveOralM   R-squared:
0.647
Model:                            OLS   Adj. R-squared:
0.640
Method:                 Least Squares   F-statistic:
91.15
Date:                Tue, 10 Sep 2024   Prob (F-statistic):
6.78e-44
Time:                        13:47:16   Log-Likelihood:
-52.991
No. Observations:                 204   AIC:
116.0
Df Residuals:                     199   BIC:
132.6
Df Model:                           4

Covariance Type:            nonrobust

======================================================================
```

```
========
                 coef    std err          t      P>|t|      [0.025
0.975]
----------------------------------------------------------------------
--------
const          8.6087      1.561      5.516      0.000       5.531
11.686
x1             1.3787      1.794      0.768      0.443      -2.159
4.917
x2            -0.8318      1.787     -0.466      0.642      -4.356
2.692
x3            -0.0998      0.086     -1.159      0.248      -0.270
0.070
x4             0.3495      0.094      3.703      0.000       0.163
0.536
======================================================================
========
Omnibus:                       25.345   Durbin-Watson:
1.857
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
41.636
Skew:                           0.684   Prob(JB):
9.09e-10
Kurtosis:                       4.740   Cond. No.
8.11e+03
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 8.11e+03. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

```
r =np.linspace(-100, 100,400)
#Definethe lossfunctions
def L1(r,a):
  return r**2/ (a**2 +r**2)

def L2(r,a):
 return 1-np.exp(-2 *np.abs(r) / a)

# Define different values of a

a_values = [2, 5, 10, 20, 30, 50, 100]

# Plotting

plt.figure(figsize=(14, 6))
```

```
# Plot L1(r) for different values of a
plt.subplot(1, 2, 1)
for a in a_values:
  plt.plot(r, L1(r, a), label=f'a = {a}')
plt.title('$L_1(r)$')
plt.xlabel('Residual $r$')
plt.ylabel('$L_1(r)$')
plt.legend()
plt.grid(True)
# Plot L2(r) for different values of a
plt.subplot(1, 2, 2)
for a in a_values:
  plt.plot(r, L2(r, a), label=f'a = {a}')
plt.title('$L_2(r)$')
plt.xlabel('Residual $r$')
plt.ylabel('$L_2(r)$')
plt.legend()
plt.grid(True)
```