pattern-assignment1-210031h

September 7, 2024

0.0.1 Assignment 01 - Learning From Data and Related Challenges and Linear Models for Regression

Import libraries

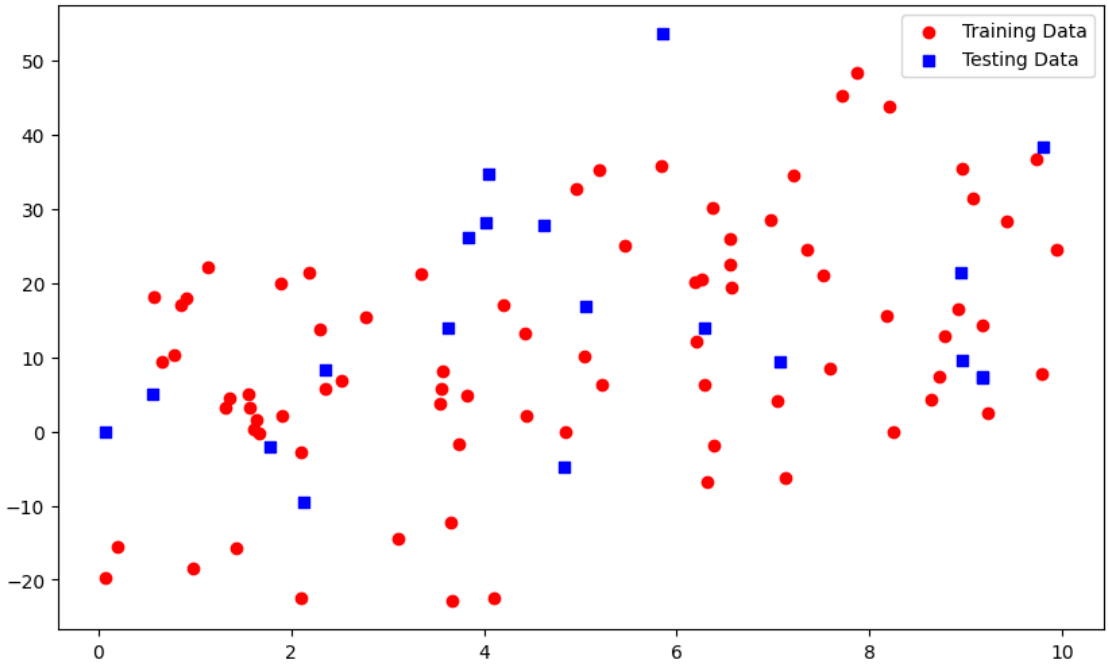
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import pandas as pd
import statsmodels.api as sm
```

Generate random data

[2]:
```
# 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]
```

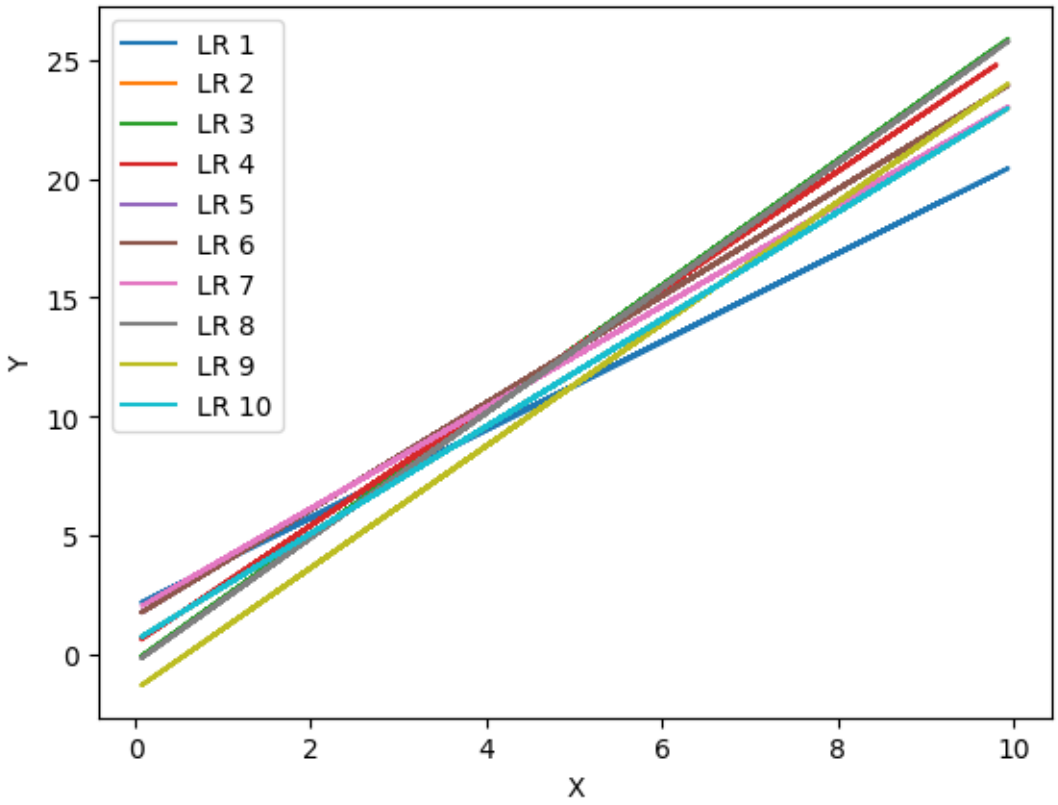
Train test split and plot

[3]:
```
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.legend()
plt.show()
```



Fit a linear regeression model

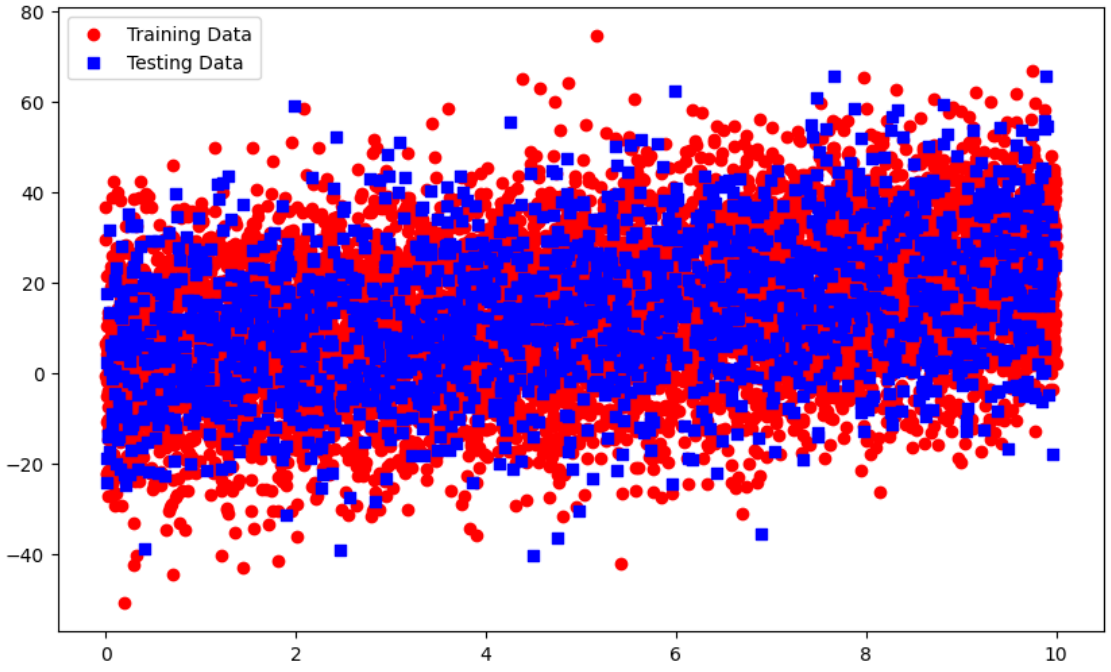
[4]:
```
for i in range(10): # Plotting 10 different instances
        X_train, X_test, Y_train, Y_test = train_test_split(X,
        Y, test_size=0.2, random_state=np.random.randint
        (104))
        #build model
        model = LinearRegression()
        #train model
        model.fit(X_train, 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()
```



Repeating with 10000 samples

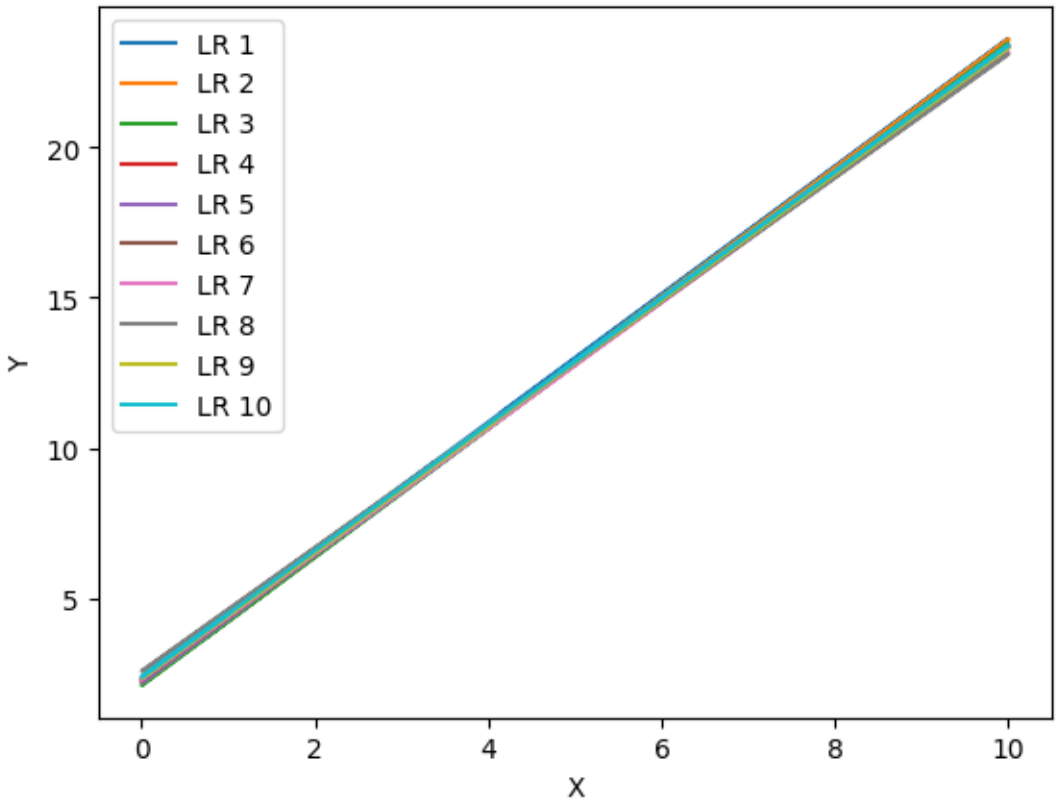
[5]:
```
# Generate 10000 samples
n_samples = 10000
# Generate X values (uniformly distributed between 0 and 10)
X = 10 * np.random.rand(n_samples, 1)
# Generate epsilon values (normally distributed with mean 0 and standard␣
 ↪deviation 15)
epsilon = np.random.normal(0, 15, n_samples)
# Generate Y values 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.legend()
plt.show()
```



LinearRegression model

[6]:
```
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))
        #build model
        model = LinearRegression()
        #train model
        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()
```



Linear Regression on real world data

[72]:
```
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

print(f"Number of independent features: {X.shape[1]}")
print(f"Number of dependent features: {Y.shape[1]}")

# metadata
print(infrared_thermography_temperature.metadata)
# variable information
print(infrared_thermography_temperature.variables)
```

```
Number of independent features: 33
Number of dependent features: 1
{'uci_id': 925, 'name': 'Infrared Thermography Temperature', 'repository_url': '
https://archive.ics.uci.edu/dataset/925/infrared+thermography+temperature+datase
t', '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/443b9932d295ca3a014e7d874b4bd77a33a276bd', '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
```

Explore dataset

[8]:
```
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')}
```

[9]:
```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1020 entries, 0 to 1019
```

```
Data columns (total 33 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Gender       1020 non-null   object
 1   Age          1020 non-null   object
 2   Ethnicity    1020 non-null   object
 3   T_atm        1020 non-null   float64
 4   Humidity     1020 non-null   float64
 5   Distance     1018 non-null   float64
 6   T_offset1    1020 non-null   float64
 7   Max1R13_1    1020 non-null   float64
 8   Max1L13_1    1020 non-null   float64
 9   aveAllR13_1  1020 non-null   float64
 10  aveAllL13_1  1020 non-null   float64
 11  T_RC1        1020 non-null   float64
 12  T_RC_Dry1    1020 non-null   float64
 13  T_RC_Wet1    1020 non-null   float64
 14  T_RC_Max1    1020 non-null   float64
 15  T_LC1        1020 non-null   float64
 16  T_LC_Dry1    1020 non-null   float64
 17  T_LC_Wet1    1020 non-null   float64
 18  T_LC_Max1    1020 non-null   float64
 19  RCC1         1020 non-null   float64
 20  LCC1         1020 non-null   float64
 21  canthiMax1   1020 non-null   float64
 22  canthi4Max1  1020 non-null   float64
 23  T_FHCC1      1020 non-null   float64
 24  T_FHRC1      1020 non-null   float64
 25  T_FHLC1      1020 non-null   float64
 26  T_FHBC1      1020 non-null   float64
 27  T_FHTC1      1020 non-null   float64
 28  T_FH_Max1    1020 non-null   float64
 29  T_FHC_Max1   1020 non-null   float64
 30  T_Max1       1020 non-null   float64
 31  T_OR1        1020 non-null   float64
 32  T_OR_Max1    1020 non-null   float64
dtypes: float64(30), object(3)
memory usage: 263.1+ KB
```

[10]:
```
y.head()
```

[10]:

	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

[31]:
```
print(f"X shape before removal: {X.shape}")
print(f"y shape before removal: {y.shape}")

# Combine X and y into a single DataFrame
data = pd.concat([X, y], axis=1)

# Drop rows with missing values
data = data.dropna()

# Split back into X and y
X_cleaned = data.iloc[:, :-1]
y_cleaned = data.iloc[:, -1]

print(f"X shape after removal:{X.shape}")
print(f"y shape after removal:{y.shape}")
```

```
X shape before removal: (1018, 33)
y shape before removal: (1018, 2)
X shape after removal:(1018, 33)
y shape after removal:(1018, 2)
```

select features

[26]:
```
# Step 1: Select the dependent feature 'aveOralM'
y_selected = y['aveOralM']

# Step 2: Select the independent features
# Choosing 'Age' and four other features ('T_atm', 'Humidity', 'Distance',␣
 ↪'Max1R13_1') based on preference
X_selected = X[['Age', 'T_atm' , 'Humidity', 'Distance', 'T_LC1']]
```

one hot encoder

[36]:
```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

# Initialize OneHotEncoder
ohe = OneHotEncoder(sparse_output=False)

# Fit and Transform the 'Age' column to get the encoded data
age_encoded = ohe.fit_transform(X_selected[['Age']])

# Convert the encoded data into a pandas DataFrame
age_encoded_df = pd.DataFrame(age_encoded, columns=ohe.
 ↪get_feature_names_out(['Age']))

# Reset index to ensure both DataFrames align correctly
age_encoded_df.reset_index(drop=True, inplace=True)
```

```
X_selected = X_selected.reset_index(drop=True)

# Combine the one-hot encoded 'Age' with the other selected features
X_encoded = pd.concat([age_encoded_df, X_selected.drop('Age', axis=1)], axis=1)

# Display the updated DataFrame
print(X_encoded.head())
print(X_encoded.shape)
```

```
   Age_18-20  Age_21-25  Age_21-30  Age_26-30  Age_31-40  Age_41-50  \
0        0.0        0.0        0.0        0.0        0.0        1.0
1        0.0        0.0        0.0        0.0        1.0        0.0
2        0.0        0.0        1.0        0.0        0.0        0.0
3        0.0        0.0        1.0        0.0        0.0        0.0
4        1.0        0.0        0.0        0.0        0.0        0.0

   Age_51-60  Age_>60  T_atm  Humidity  Distance    T_LC1
0        0.0      0.0   24.0      28.0       0.8  35.3375
1        0.0      0.0   24.0      26.0       0.8  34.5600
2        0.0      0.0   24.0      26.0       0.8  35.5025
3        0.0      0.0   24.0      27.0       0.8  35.5950
4        0.0      0.0   24.0      27.0       0.8  35.6400
(1018, 12)
```

linear regression model

[48]:
```
from sklearn import model_selection as  model_selection

X_train, X_test, y_train, y_test = model_selection.train_test_split(X_encoded,␣
 ↪y_selected, test_size=0.2, random_state=0)

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Initialize the Linear Regression model
model = LinearRegression()

# Train the model using the training data
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

print('Intercept:\n',model.intercept_)
```

```
Intercept:
 11.729367368809324
```

get coefficients

[70]:
```
# Get the estimated coefficients
coefficients = model.coef_

# Create a DataFrame to display the coefficients with corresponding feature␣
 ↪names
coef_df = pd.DataFrame(coefficients, X_train.columns, columns=['Estimated␣
 ↪Coefficients'])

# Display the estimated coefficients
print(coef_df)
```

```
            Estimated Coefficients
T_OR1                     0.503390
T_OR_Max1                 0.021690
T_FHC_Max1               -0.060225
T_FH_Max1                 0.359363
```

model with different features

[60]:
```
# Select the dependent feature 'aveOralM'
y_selected = data['aveOralM']

# Select the independent features
X_selected = data[['T_OR1', 'T_OR_Max1', 'T_FHC_Max1', 'T_FH_Max1' ]]

X_train, X_test, y_train, y_test = model_selection.train_test_split(X_selected,␣
 ↪y_selected, test_size=0.2, random_state=0)

# Initialize the Linear Regression model
model = LinearRegression()

# Train the model using the training data
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

model.score(X_test, y_test)

print(model.coef_)  # Coefficients
print(model.intercept_)  # Intercept
```

```
[ 0.50338995  0.02169014 -0.06022472  0.35936308]
7.611464539155499
```

calculate errors

[66]:
```
# Calculate Residual Sum of Squares (RSS)
RSS = np.sum((np.array(y_test) - y_pred) ** 2)
print("RSS:", RSS)

# Calculate Residual Standard Error (RSE)
n = len(y_test)  # Number of observations
p = 4  # Number of features
RSE = np.sqrt(RSS / (n - p - 1))
print("RSE", RSE)

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)
print(f"MSE:",mse)

# Calculate R^2 Score
r2 = r2_score(y_test, y_pred)
print(f"R^2:",r2)
```

```
RSS: 18.80409309153579
RSE 0.3073970235838663
MSE: 0.09217692691929309
R^2: 0.6495141698846425
```

[67]:
```
X_sm = sm.add_constant(X_test)

# Fit the model using statsmodels
model_sm = sm.OLS(y_test, X_sm).fit()

# Get standard errors for each feature
standard_errors = model_sm.bse
print("Standard Errors for Each Feature:", standard_errors[1:],sep="\n")
```

```
Standard Errors for Each Feature:
T_OR1         1.831723
T_OR_Max1     1.827018
T_FHC_Max1    0.089940
T_FH_Max1     0.094565
dtype: float64
```

[68]:
```
# Get t-statistics for each feature
t_statistics = model_sm.tvalues
print("t-statistics for Each Feature:", t_statistics[1:],sep='\n')
```

```
t-statistics for Each Feature:
T_OR1        -0.371945
T_OR_Max1     0.716412
T_FHC_Max1   -0.728860
T_FH_Max1     2.996086
```

```
dtype: float64
```

[69]:
```
# Get p-values for each feature
p_values = model_sm.pvalues
print("p-values for Each Feature:", p_values[1:],sep='\n')
```

```
p-values for Each Feature:
T_OR1          0.710330
T_OR_Max1      0.474577
T_FHC_Max1     0.466944
T_FH_Max1      0.003082
dtype: float64
```