```
# importing modules and paskages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn import preprocessing

from google.colab import files
uploaded = files.upload()
```

```
Pilih File Real-estate1.csv
• Real-estate1.csv(text/csv) - 21968 bytes, last modified: 30/10/2023 - 100% done
Saving Real-estate1.csv to Real-estate1.csv
```

```
# importing data
df = pd.read_csv('Real-estate1.csv')
df.drop('No', inplace = True,axis=1)

print(df.head())
print(df.columns)
```

```
   X1 transaction date  X2 house age  X3 distance to the nearest MRT station  \
0             2012.917          32.0                                84.87882
1             2012.917          19.5                               306.59470
2             2013.583          13.3                               561.98450
3             2013.500          13.3                               561.98450
4             2012.833           5.0                               390.56840

   X4 number of convenience stores  X5 latitude  X6 longitude  \
0                               10     24.98298     121.54024
1                                9     24.98034     121.53951
2                                5     24.98746     121.54391
3                                5     24.98746     121.54391
4                                5     24.97937     121.54245

   Y house price of unit area
0                        37.9
1                        42.2
2                        47.3
3                        54.8
4                        43.1
Index(['X1 transaction date', 'X2 house age',
       'X3 distance to the nearest MRT station',
       'X4 number of convenience stores', 'X5 latitude', 'X6 longitude',
       'Y house price of unit area'],
      dtype='object')
```

```
# plotting a scatterplot
sns.scatterplot(x='X4 number of convenience stores',
                y='Y house price of unit area', data=df)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-c898549ce074> in <cell line: 2>()
      1 # plotting a scatterplot
      2 sns.scatterplot(x='X4 number of convenience stores',
----> 3                 y='Y house price of unit area', data=df)

NameError: name 'df' is not defined
```

TELUSURI STACK OVERFLOW

```
# creating feature variables
X = df.drop('Y house price of unit area', axis= 1)
y = df['Y house price of unit area']
print(x)
print(y)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-15-d2b1227c8130> in <cell line: 2>()
      1 # creating feature variables
----> 2 X = df.drop('X house price of unit area', axis= 1)
      3 y = df['Y house price of unit area']
      4 print(x)
      5 print(y)

                                5 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in drop(self, labels, errors)
   6932         if mask.any():
```

```
# creating train anda test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

```
   6936         return self.delete(indexer)
```

```
# creating a regression model
model = LinearRegression()
```

TELUSURI STACK OVERFLOW

```
# fitting the model
model.fit(X_train,y _train)
```

```
  File "<ipython-input-19-05e9bb32a7b9>", line 2
    model.fit(X_train,y _train)
                       ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

TELUSURI STACK OVERFLOW

```
# making predictions
predictions = model.predict(X_test)
```

```
---------------------------------------------------------------------------
NotFittedError                            Traceback (most recent call last)
<ipython-input-20-6dd31a133758> in <cell line: 2>()
      1 # making predictions
----> 2 predictions = model.predict(X_test)

                                2 frames
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py in check_is_fitted(estimator, attributes, msg, all_or_any)
   1388
   1389     if not fitted:
-> 1390         raise NotFittedError(msg % {"name": type(estimator).__name__})
   1391
   1392

NotFittedError: This LinearRegression instance is not fitted yet. Call 'fit' with appropriate arguments before using this estimator.
```

TELUSURI STACK OVERFLOW

```
# model evaluation
print(
    'mean_squared_error : ', mean_squared_error(y_test, predictions))
print(
    'mean_absolute_error : ', mean_absolute_error(y_test, predictions))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-7-4224b0c2529d> in <cell line: 2>()
      1 # model evaluation
      2 print(
----> 3     'mean_squared_error : ', mean_squared_error(y_test, predictions))
      4 print(
      5     'mean_absolute_error : ', mean_absolute_error(y_test, predictions))

NameError: name 'predictions' is not defined
```

TELUSURI STACK OVERFLOW

```
# import libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import numpy as np

# load the iris dataset
```

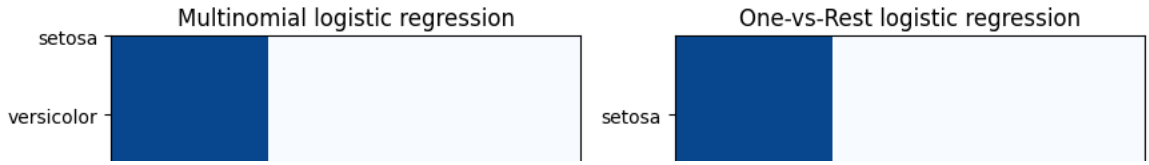
```
iris = load_iris()
x = iris.data
y = iris.target

# split the data into training and testing sets
x_train, x_test,\
y_train, y_test = train_test_split(x, y,
                                   test_size=0.2,
                                   random_state=42)

# create a Multinomial logistic regression model
multi_logreg = LogisticRegression(multi_class='multinomial',
                                  solver='lbfgs')
multi_logreg.fit(x_train, y_train)

# create a One-vs-Rest logistic regression model
ovr_logreg = LogisticRegression(multi_class='ovr',
                                solver='liblinear')
ovr_logreg.fit(x_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

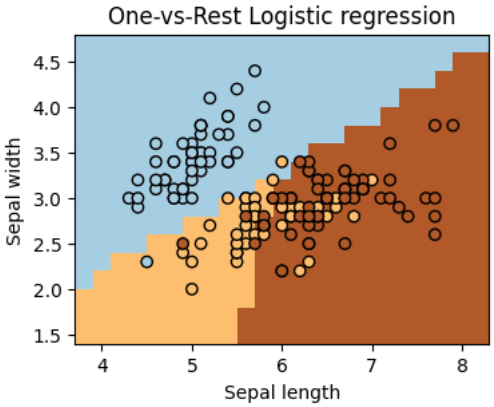
```
LogisticRegression
LogisticRegression(multi_class='ovr', solver='liblinear')
```

```
# make predictions using the trained models
y_pred_multi = multi_logreg.predict(x_test)
y_pred_ovr = ovr_logreg.predict(x_test)

# evaluate the performance of the models
# using accuracy score and confusion matrix
print('Multinomial logistic regression accuracy:',
      accuracy_score(y_test, y_pred_multi))
print('One-vs-Rest logistic regression accuracy:',
      accuracy_score(y_test, y_pred_ovr))

conf_mat_multi = confusion_matrix(y_test, y_pred_multi)
conf_mat_ovr = confusion_matrix(y_test, y_pred_ovr)

# plot the confusion matrices
fig, axs = plt.subplots(ncols=2, figsize=(10, 5))
axs[0].imshow(conf_mat_multi, cmap=plt.cm.Blues)
axs[0].set_title('Multinomial logistic regression')
axs[0].set_xlabel('Predicted labels')
axs[0].set_ylabel('True labels')
axs[0].set_xticks(np.arange(len(iris.target_names)))
axs[0].set_xticklabels(iris.target_names)
axs[0].set_yticklabels(iris.target_names)
axs[1].imshow(conf_mat_ovr, cmap=plt.cm.Blues)
axs[1].set_title('One-vs-Rest logistic regression')
axs[1].set_xlabel('Predicted labels')
axs[1].set_ylabel('True labels')
axs[1].set_xticks(np.arange(len(iris.target_names)))
axs[1].set_xticklabels(iris.target_names)
axs[1].set_yticks(np.arange(len(iris.target_names)))
axs[1].set_yticklabels(iris.target_names)
plt.show()
```

```
Multinomial logistic regression accuracy: 1.0
One-vs-Rest logistic regression accuracy: 1.0
<ipython-input-5-aa80e91df5ca>:23: UserWarning: FixedFormatter should only be used together with FixedLocator
  axs[0].set_yticklabels(iris.target_names)
```




```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression

iris=load_iris()

X=iris.data[:,:2]
y=iris.target

clf=LogisticRegression(random_state=0,
                       multi_class='ovr',
                       solver='liblinear')

clf.fit(X,y)

x_min, x_max=X[:,0].min()-.5,X[:,0].max()+.5
y_min, y_max=X[:,1].min()-.5,X[:,1].max()+.5
xx,yy=np.meshgrid(np.arange(x_min,x_max,0.2),
                  np.arange(y_min,y_max,0.2))
Z=clf.predict(np.c_[xx.ravel(),yy.ravel()])
Z=Z.reshape(xx.shape)
plt.figure(1,figsize=(4,3))
plt.pcolormesh(xx,yy,Z,cmap=plt.cm.Paired)
plt.scatter(X[:,0],X[:,1],c=y,edgecolors='k',
            cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')
plt.title('One-vs-Rest Logistic regression')
plt.show()
```




```
# create a Multinominal logistic regression model
multi_logreg = LogisticRegression(multi_class='multinomial',
                                  solver='lbfgs')
multi_logreg.fit(X_train, y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-25-c808734dc3d5> in <cell line: 4>()
```

```
# create a One-vs-Rest logistic regression model
ovr_logreg = LogisticRegression(multi_class='ovr',
                                solver='liblinear')
ovr_logreg.fit(X_train, y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-26-082173f1aea3> in <cell line: 4>()
      2 ovr_logreg = LogisticRegression(multi_class='ovr',
      3                                 solver='liblinear')
----> 4 ovr_logreg.fit(X_train, y_train)

                                    1 frames
/usr/local/lib/python3.10/dist-packages/sklearn/utils/multiclass.py in check_classification_targets(y)
    216         "multilabel-sequences",
    217     ]:
--> 218         raise ValueError("Unknown label type: %r" % y_type)
    219
    220

ValueError: Unknown label type: 'continuous'
```

TELUSURI STACK OVERFLOW