❖ Day 4: Supervised Learning Algorithms and Introduction to Linear Regression

❖ Supervised Machine Learning:

Supervised machine learning learns patterns and relationships between input and output data. It is defined by its use of labeled data.

- There are two types of supervised learning algorithms:
    1. Classification
    2. Regression
- 1. Classification:

  Classification is a supervised machine learning method where the model tries to predict the correct label of a given input data.
  - There are many machine learning algorithms that can be used for classification tasks. Some of them are:

    I. Logistic Regression

    II. Decision Tree Classifier

    III. K Nearest Neighbor Classifier

    IV. Random Forest Classifier

I. Logistic Regression:

Logistic Regression is a special case of Linear Regression where target variable (y) is discrete / categorical such as 1 or 0, True or False, Yes or No, Default or No Default.

- Advantage:

  -Simple and interpretable.

  - Linear decision boundary.

- Disadvantage:
  - Assumes linearity.
  - Limited to binary classification.

II. Decision Tree:

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks.

- Advantage:

  - Interpretability.

  - No assumptions about data distribution.

- Disadvantage:
  - Prone to overfitting.
  - Bias toward dominant classes.

III. K Nearest Neighbor Classifier:

K-Nearest Neighbors is a statistical method that evaluates the proximity of one data point to another data point in order to decide whether or not the two data points can be grouped together.

- Advantage:
  - Simple and intuitive.
  - No training period.

- Disadvantage:
  - Computational complexity.
  - Sensitivity to irrelevant features.

V. Random Forest Classifier:

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.

- Advantage:
  - High accuracy.
  - Reduced overfitting.
- Disadvantage:
  - Computational complexity.
  - Less interpretability.

2.Regression:

Regression shows a line or curve that passes through all the datapoints on target-predictor graph in such a way that the vertical distance between the datapoints and the regression line is minimum.

- There are many machine learning algorithms that can be used for regression tasks. Some of them are:

  I. Linear regression:

II. Polynomial regression

III. Lasso regression

I. Linear regression:

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis.

We can represented of linear regression:

$y= a_0+a_1x+ \varepsilon$

- Types of linear regression:
    1. Single linear regression
    2. Multiple linear regression
- Advantage:

  - Simplicity.

  - Interpretability.

- Disadvantage:

  - Assumption of linearity.

  - Sensitive to outliers.

II. Polynomial Regression:

Polynomial Regression is a regression algorithm that models the relationship between a dependent(y) and independent variable(x) as nth degree polynomial.

- The Polynomial Regression equation is given below:

$y= b_0+b_1x_1+ b_2x_1^2+ b_2x_1^3+..... b_nx_1^n$

- Advantage:

  - Flexibility for non-linear relationships.

  - Captures complex patterns.

- Disadvantage:

  - Prone to overfitting.

  - Reduced interpretability with higher degrees.

III. Lasso Regression:

Lasso is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model.

- Advantage:

  - Feature selection.

  - Handles multicollinearity.

- Disadvantage:

  - Unstable for highly correlated features.

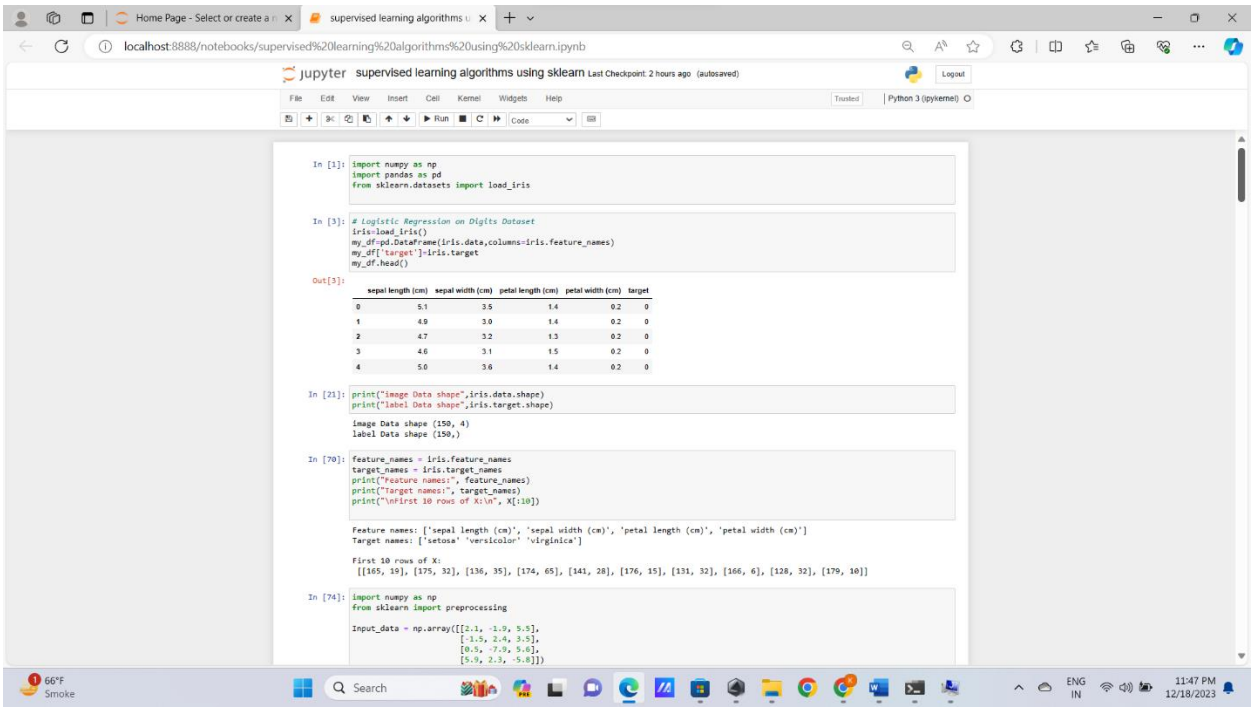
  - Sensitive to outliers.

- Command:

  Pip install scikit-learn

❖Description:

I'm thrilled to apply this newfound knowledge to real-world scenarios, and the prospect of further honing these skills through the Skill Boost Internship Program adds an extra layer of excitement. Here's to the journey ahead and the exciting challenges awaiting in the Skill Boost Internship Program(www.Batweb.com).



Home Page - Select or create a n | supervised learning algorithms u

localhost:8888/notebooks/supervised%20learning%20algorithms%20using%20sklearn.ipynb

jupyter **supervised learning algorithms using sklearn** Last Checkpoint: 2 hours ago (autosaved) — Logout

File Edit View Insert Cell Kernel Widgets Help — Trusted — Python 3 (ipykernel)

Run — Code

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
```

In [3]:
```
# Logistic Regression on Digits Dataset
iris=load_iris()
my_df=pd.DataFrame(iris.data,columns=iris.feature_names)
my_df['target']=iris.target
my_df.head()
```

Out[3]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [21]:
```
print("image Data shape",iris.data.shape)
print("label Data shape",iris.target.shape)
```

```
image Data shape (150, 4)
label Data shape (150,)
```

In [70]:
```
feature_names = iris.feature_names
target_names = iris.target_names
print("Feature names:", feature_names)
print("Target names:", target_names)
print("\nFirst 10 rows of X:\n", X[:10])
```

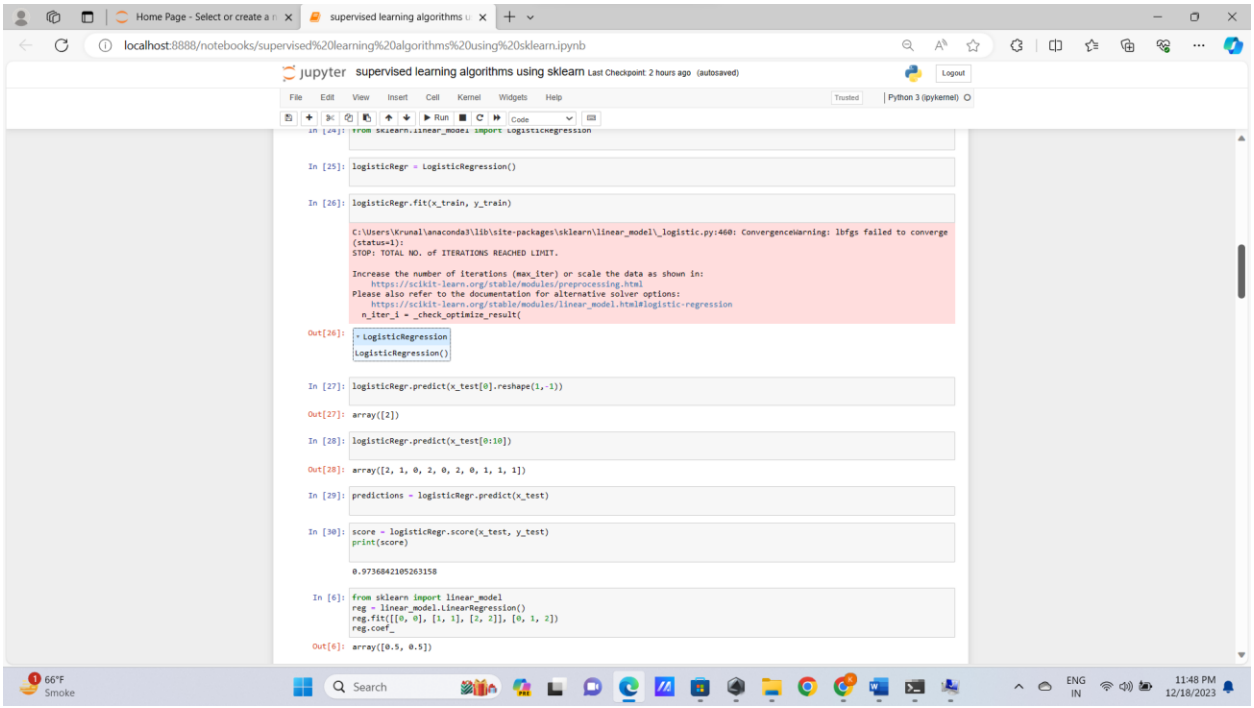
```
Feature names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Target names: ['setosa' 'versicolor' 'virginica']

First 10 rows of X:
 [[165, 19], [175, 32], [136, 35], [174, 65], [141, 28], [176, 15], [131, 32], [166, 6], [128, 32], [179, 10]]
```

In [74]:
```
import numpy as np
from sklearn import preprocessing

Input_data = np.array([[2.1, -1.9, 5.5],
                       [-1.5, 2.4, 3.5],
                       [0.5, -7.9, 5.6],
                       [5.9, 2.3, -5.8]])
```

66°F Smoke — Search — ENG IN — 11:47 PM 12/18/2023



Home Page - Select or create a n | supervised learning algorithms u

localhost:8888/notebooks/supervised%20learning%20algorithms%20using%20sklearn.ipynb

jupyter **supervised learning algorithms using sklearn** Last Checkpoint: 2 hours ago (autosaved) — Logout

File Edit View Insert Cell Kernel Widgets Help — Trusted — Python 3 (ipykernel)

Run — Code

In [24]:
```
from sklearn.linear_model import LogisticRegression
```

In [25]:
```
logisticRegr = LogisticRegression()
```

In [26]:
```
logisticRegr.fit(x_train, y_train)
```

```
C:\Users\Krunal\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge
(status=1):
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(
```

Out[26]:
```
LogisticRegression()
```

In [27]:
```
logisticRegr.predict(x_test[0].reshape(1,-1))
```

Out[27]:
```
array([2])
```

In [28]:
```
logisticRegr.predict(x_test[0:10])
```

Out[28]:
```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1])
```

In [29]:
```
predictions = logisticRegr.predict(x_test)
```

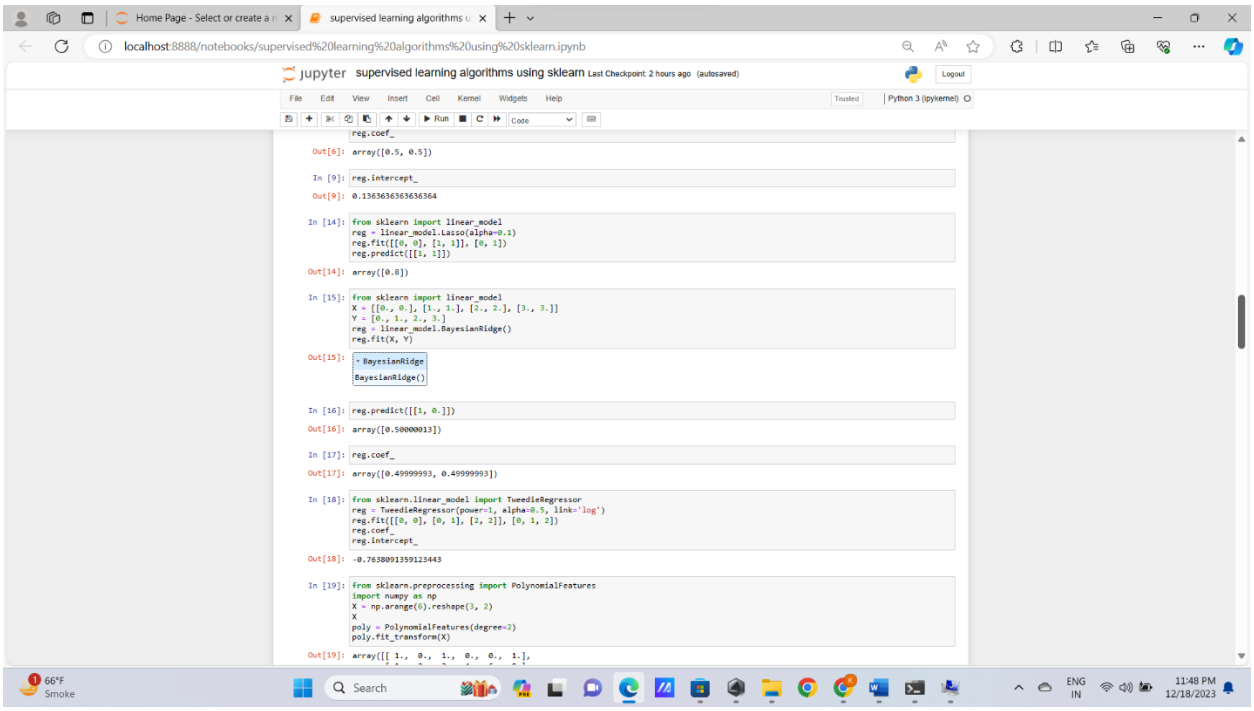
In [30]:
```
score = logisticRegr.score(x_test, y_test)
print(score)
```

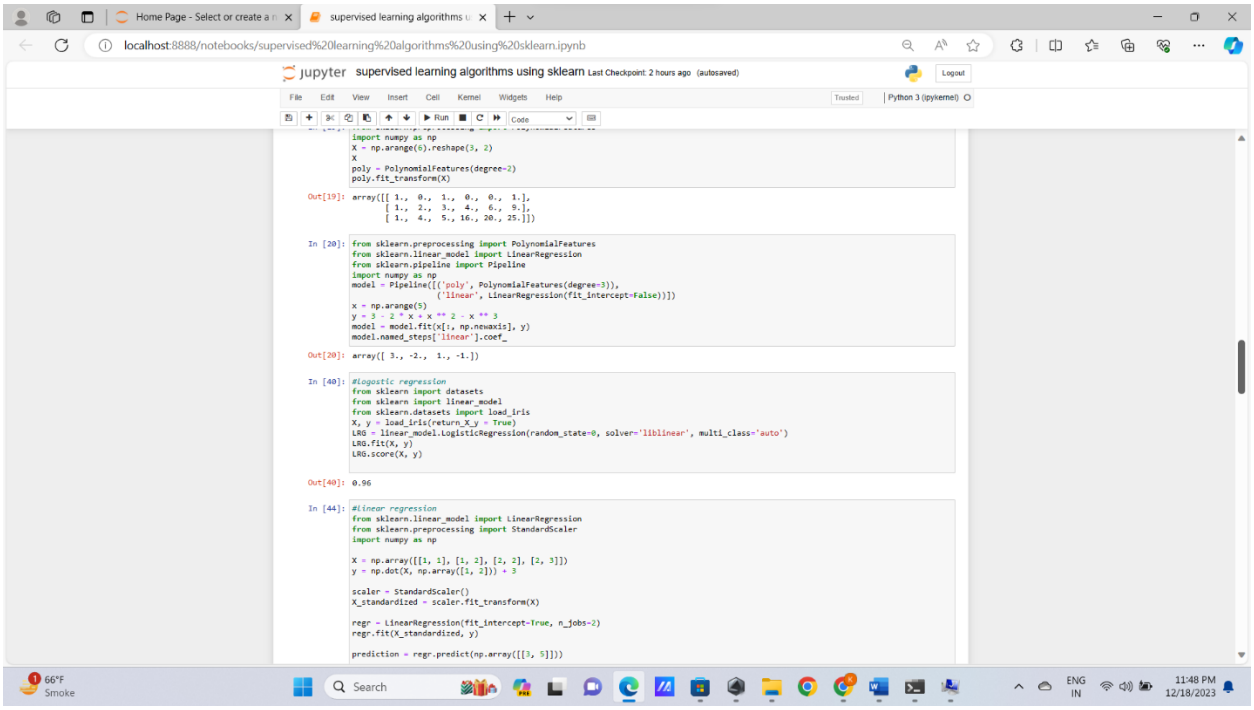
```
0.9736842105263158
```

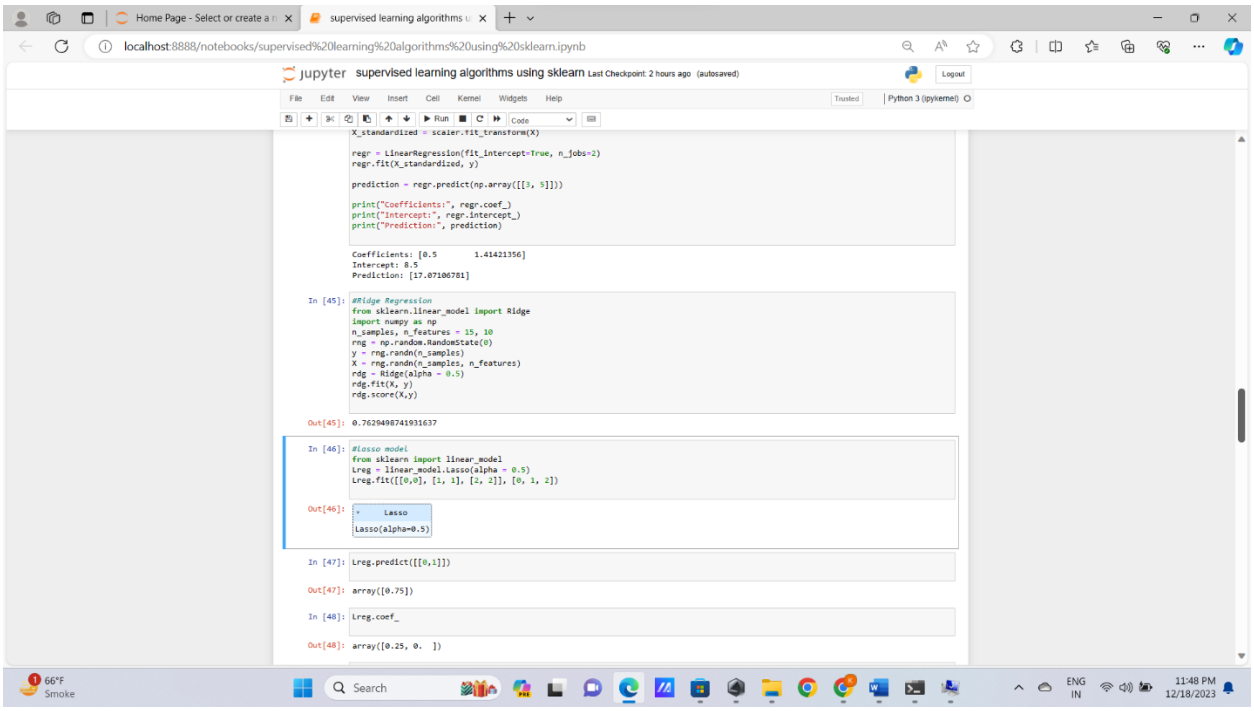
In [6]:
```
from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
reg.coef_
```

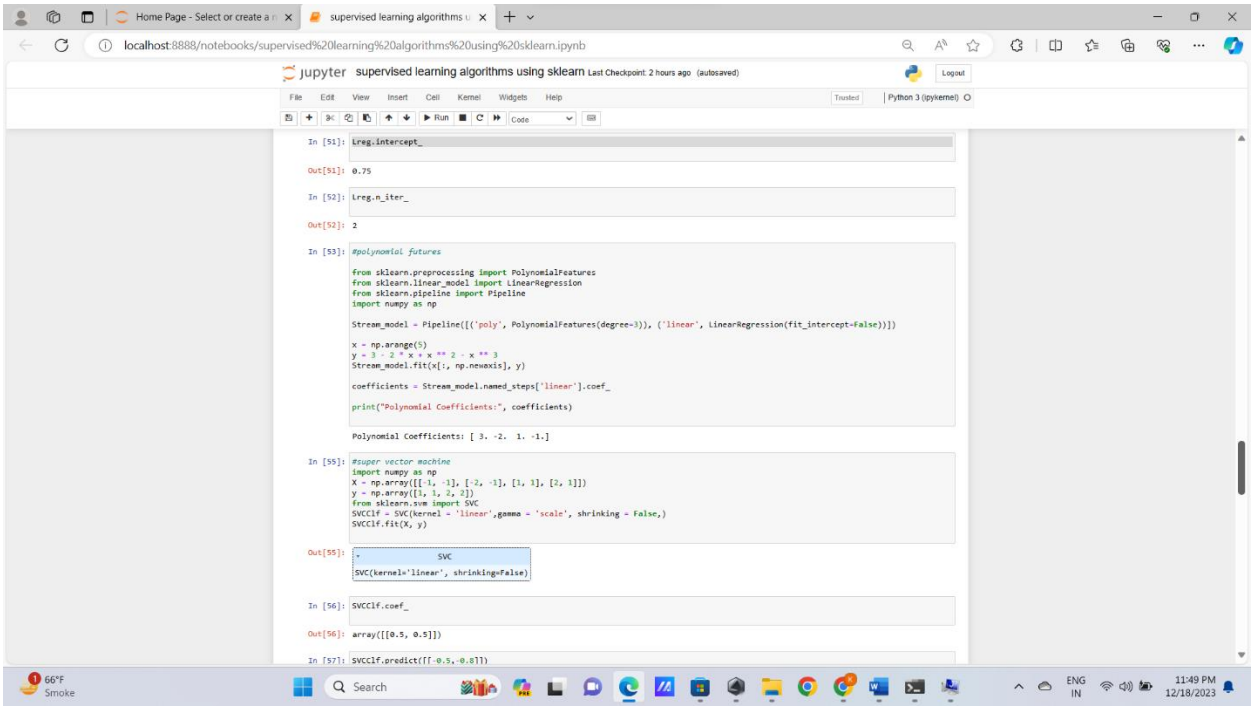
Out[6]:
```
array([0.5, 0.5])
```

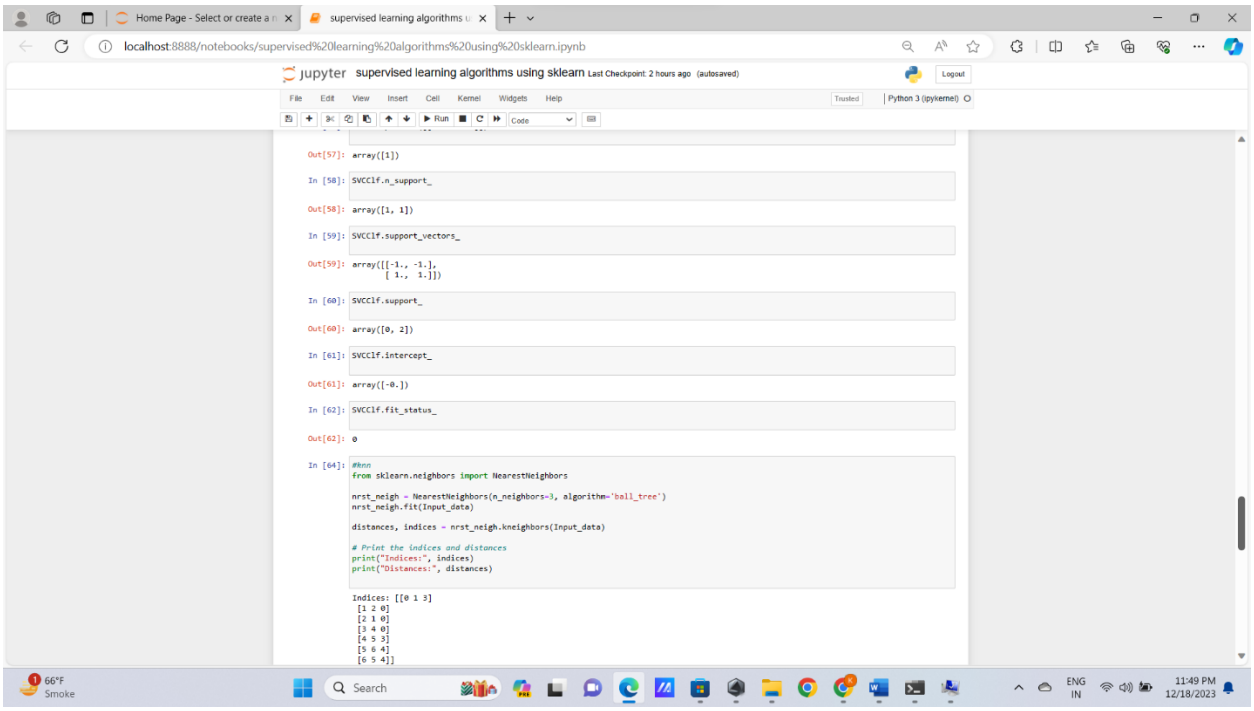
66°F Smoke — Search — ENG IN — 11:48 PM 12/18/2023

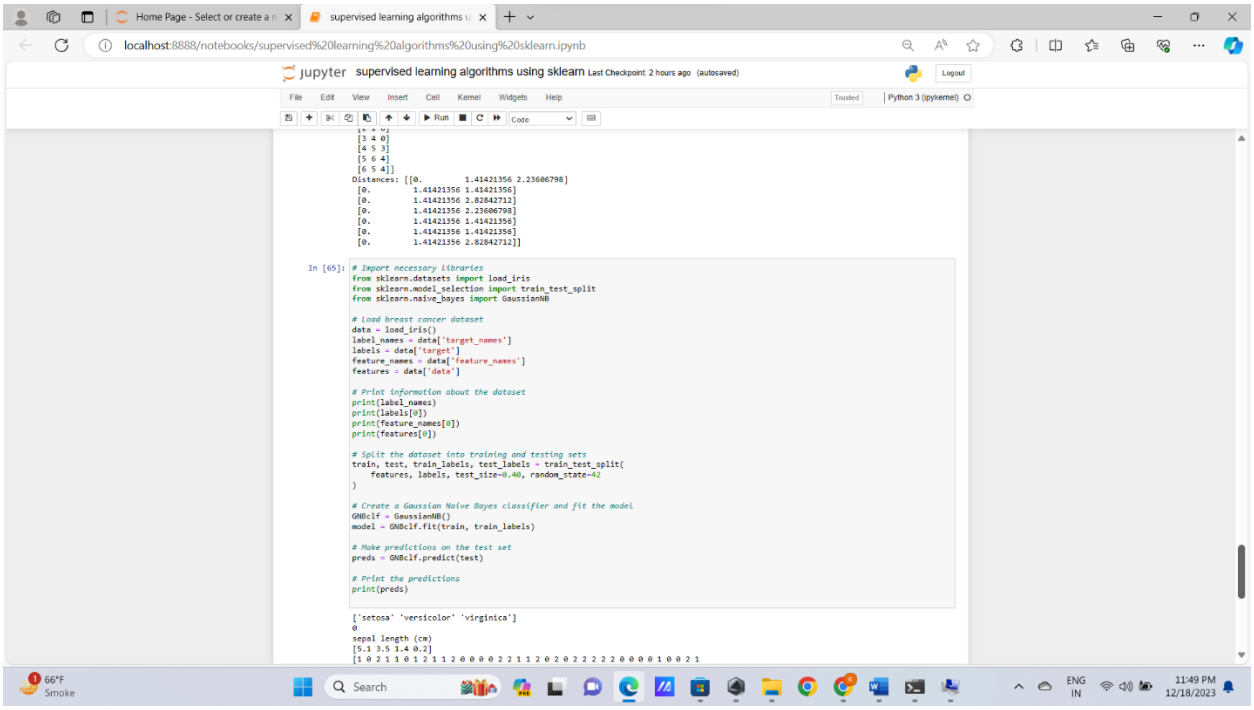

Home Page - Select or create a n
supervised learning algorithms u
localhost:8888/notebooks/supervised%20learning%20algorithms%20using%20sklearn.ipynb
jupyter supervised learning algorithms using sklearn Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
reg.coef_
Out[6]: array([0.5, 0.5])
In [9]: reg.intercept_
Out[9]: 0.13636363636363664
In [14]: from sklearn import linear_model
reg = linear_model.Lasso(alpha=0.1)
reg.fit([[0, 0], [1, 1]], [0, 1])
reg.predict([[1, 1]])
Out[14]: array([0.8])
In [15]: from sklearn import linear_model
X = [[0., 0.], [1., 1.], [2., 2.], [3., 3.]]
Y = [0., 1., 2., 3.]
reg = linear_model.BayesianRidge()
reg.fit(X, Y)
Out[15]: BayesianRidge
BayesianRidge()
In [16]: reg.predict([[1, 0.]])
Out[16]: array([0.50000013])
In [17]: reg.coef_
Out[17]: array([0.49999993, 0.49999993])
In [18]: from sklearn.linear_model import TweedieRegressor
reg = TweedieRegressor(power=1, alpha=0.5, link='log')
reg.fit([[0, 0], [0, 1], [2, 2]], [0, 1, 2])
reg.coef_
reg.intercept_
Out[18]: -0.7638091359123443
In [19]: from sklearn.preprocessing import PolynomialFeatures
import numpy as np
X = np.arange(6).reshape(3, 2)
X
poly = PolynomialFeatures(degree=2)
poly.fit_transform(X)
Out[19]: array([[1., 0., 1., 0., 0., 1.],
66°F
Smoke
Search
ENG
IN
11:48 PM
12/18/2023

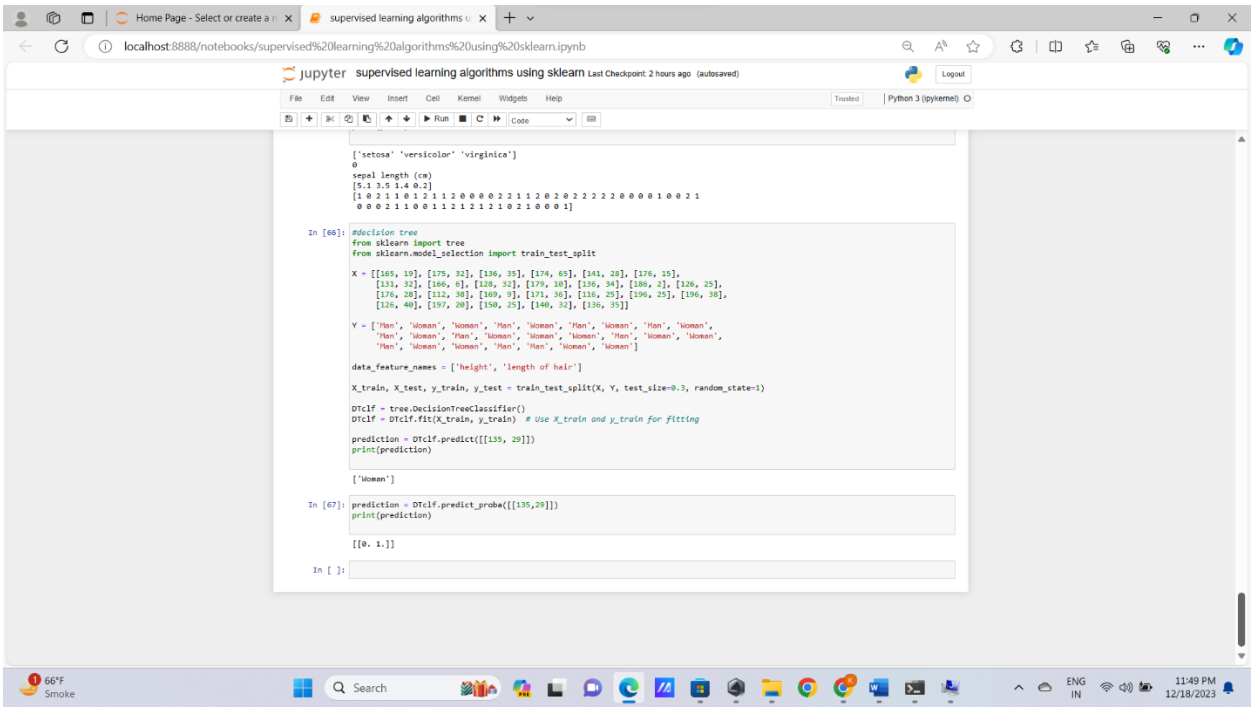

Home Page - Select or create a n
supervised learning algorithms u
localhost:8888/notebooks/supervised%20learning%20algorithms%20using%20sklearn.ipynb
jupyter supervised learning algorithms using sklearn Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
import numpy as np
X = np.arange(6).reshape(3, 2)
X
poly = PolynomialFeatures(degree=2)
poly.fit_transform(X)
Out[19]: array([[1., 0., 1., 0., 0., 1.],
[1., 2., 3., 4., 6., 9.],
[1., 4., 5., 16., 20., 25.]])
In [20]: from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
import numpy as np
model = Pipeline([('poly', PolynomialFeatures(degree=3)),
('linear', LinearRegression(fit_intercept=False))])
x = np.arange(5)
y = 3 - 2 * x + x ** 2 - x ** 3
model = model.fit(x[:, np.newaxis], y)
model.named_steps['linear'].coef_
Out[20]: array([3., -2., 1., -1.])
In [40]: #Logostic regression
from sklearn import datasets
from sklearn import linear_model
from sklearn.datasets import load_iris
X, y = load_iris(return_X_y = True)
LRG = linear_model.LogisticRegression(random_state=0, solver='liblinear', multi_class='auto')
LRG.fit(X, y)
LRG.score(X, y)
Out[40]: 0.96
In [44]: #Linear regression
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
import numpy as np
X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
y = np.dot(X, np.array([1, 2])) + 3
scaler = StandardScaler()
X_standardized = scaler.fit_transform(X)
regr = LinearRegression(fit_intercept=True, n_jobs=2)
regr.fit(X_standardized, y)
prediction = regr.predict(np.array([[3, 5]]))
66°F
Smoke
Search
ENG
IN
11:48 PM
12/18/2023


Home Page - Select or create a n
supervised learning algorithms u
localhost:8888/notebooks/supervised%20learning%20algorithms%20using%20sklearn.ipynb
jupyter supervised learning algorithms using sklearn Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
X_standardized = scaler.fit_transform(X)
regr = LinearRegression(fit_intercept=True, n_jobs=2)
regr.fit(X_standardized, y)
prediction = regr.predict(np.array([[3, 5]]))
print("Coefficients:", regr.coef_)
print("Intercept:", regr.intercept_)
print("Prediction:", prediction)
Coefficients: [0.5 1.41421356]
Intercept: 8.5
Prediction: [17.07106781]
In [45]:
#Ridge Regression
from sklearn.linear_model import Ridge
import numpy as np
n_samples, n_features = 15, 10
rng = np.random.RandomState(0)
y = rng.randn(n_samples)
X = rng.randn(n_samples, n_features)
rdg = Ridge(alpha = 0.5)
rdg.fit(X, y)
rdg.score(X,y)
Out[45]: 0.7629498741931637
In [46]:
#Lasso model
from sklearn import linear_model
Lreg = linear_model.Lasso(alpha = 0.5)
Lreg.fit([[0,0], [1, 1], [2, 2]], [0, 1, 2])
Out[46]:
Lasso
Lasso(alpha=0.5)
In [47]:
Lreg.predict([[0,1]])
Out[47]: array([0.75])
In [48]:
Lreg.coef_
Out[48]: array([0.25, 0.])
66°F
Smoke
Search
ENG
IN
11:48 PM
12/18/2023


Home Page - Select or create a n
supervised learning algorithms u
localhost:8888/notebooks/supervised%20learning%20algorithms%20using%20sklearn.ipynb
jupyter supervised learning algorithms using sklearn Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [51]:
Lreg.intercept_
Out[51]: 0.75
In [52]:
Lreg.n_iter_
Out[52]: 2
In [53]:
#polynomial futures
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
import numpy as np
Stream_model = Pipeline([('poly', PolynomialFeatures(degree=3)), ('linear', LinearRegression(fit_intercept=False))])
x = np.arange(5)
y = 3 - 2 * x + x ** 2 - x ** 3
Stream_model.fit(x[:, np.newaxis], y)
coefficients = Stream_model.named_steps['linear'].coef_
print("Polynomial Coefficients:", coefficients)
Polynomial Coefficients: [3. -2. 1. -1.]
In [55]:
#super vector machine
import numpy as np
X = np.array([[-1, -1], [-2, -1], [1, 1], [2, 1]])
y = np.array([1, 1, 2, 2])
from sklearn.svm import SVC
SVCClf = SVC(kernel = 'linear',gamma = 'scale', shrinking = False,)
SVCClf.fit(X, y)
Out[55]:
SVC
SVC(kernel='linear', shrinking=False)
In [56]:
SVCClf.coef_
Out[56]: array([[0.5, 0.5]])
In [57]:
SVCClf.predict([[-0.5,-0.8]])
66°F
Smoke
Search
ENG
IN
11:49 PM
12/18/2023


Home Page - Select or create a n
supervised learning algorithms u
localhost:8888/notebooks/supervised%20learning%20algorithms%20using%20sklearn.ipynb
jupyter supervised learning algorithms using sklearn Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Out[57]: array([1])
In [58]: SVCClf.n_support_
Out[58]: array([1, 1])
In [59]: SVCClf.support_vectors_
Out[59]: array([[-1., -1.],
[1., 1.]])
In [60]: SVCClf.support_
Out[60]: array([0, 2])
In [61]: SVCClf.intercept_
Out[61]: array([-0.])
In [62]: SVCClf.fit_status_
Out[62]: 0
In [64]: #knn
from sklearn.neighbors import NearestNeighbors
nrst_neigh = NearestNeighbors(n_neighbors=3, algorithm='ball_tree')
nrst_neigh.fit(Input_data)
distances, indices = nrst_neigh.kneighbors(Input_data)
# Print the indices and distances
print("Indices:", indices)
print("Distances:", distances)
Indices: [[0 1 3]
[1 2 0]
[2 1 0]
[3 4 0]
[4 5 3]
[5 6 4]
[6 5 4]]
66°F
Smoke
Search
ENG
IN
11:49 PM
12/18/2023


Home Page - Select or create a n
supervised learning algorithms u
localhost:8888/notebooks/supervised%20learning%20algorithms%20using%20sklearn.ipynb
jupyter supervised learning algorithms using sklearn Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
[3 4 0]
[4 5 3]
[5 6 4]
[6 5 4]]
Distances: [[0. 1.41421356 2.23606798]
[0. 1.41421356 1.41421356]
[0. 1.41421356 2.82842712]
[0. 1.41421356 2.23606798]
[0. 1.41421356 1.41421356]
[0. 1.41421356 1.41421356]
[0. 1.41421356 2.82842712]]
In [65]: # Import necessary libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
# Load breast cancer dataset
data = load_iris()
label_names = data['target_names']
labels = data['target']
feature_names = data['feature_names']
features = data['data']
# Print information about the dataset
print(label_names)
print(labels[0])
print(feature_names[0])
print(features[0])
# Split the dataset into training and testing sets
train, test, train_labels, test_labels = train_test_split(
features, labels, test_size=0.40, random_state=42
)
# Create a Gaussian Naive Bayes classifier and fit the model
GNBclf = GaussianNB()
model = GNBclf.fit(train, train_labels)
# Make predictions on the test set
preds = GNBclf.predict(test)
# Print the predictions
print(preds)
['setosa' 'versicolor' 'virginica']
0
sepal length (cm)
[5.1 3.5 1.4 0.2]
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 2 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1
66°F
Smoke
Search
ENG
IN
11:49 PM
12/18/2023


Home Page - Select or create a n
supervised learning algorithms u
localhost:8888/notebooks/supervised%20learning%20algorithms%20using%20sklearn.ipynb
jupyter supervised learning algorithms using sklearn Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
['setosa' 'versicolor' 'virginica']
0
sepal length (cm)
[5.1 3.5 1.4 0.2]
[1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 2 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1
0 0 0 2 1 1 0 0 1 1 2 1 2 1 2 1 0 2 1 0 0 0 1]
In [66]:
#decision tree
from sklearn import tree
from sklearn.model_selection import train_test_split
X = [[165, 19], [175, 32], [136, 35], [174, 65], [141, 28], [176, 15],
[131, 32], [166, 6], [128, 32], [179, 10], [136, 34], [186, 2], [126, 25],
[176, 28], [112, 38], [169, 9], [171, 36], [116, 25], [196, 25], [196, 38],
[126, 40], [197, 20], [150, 25], [140, 32], [136, 35]]
Y = ['Man', 'Woman', 'Woman', 'Man', 'Woman', 'Man', 'Woman', 'Man', 'Woman',
'Man', 'Woman', 'Man', 'Woman', 'Woman', 'Woman', 'Man', 'Woman', 'Woman',
'Man', 'Woman', 'Woman', 'Man', 'Man', 'Woman', 'Woman']
data_feature_names = ['height', 'length of hair']
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=1)
DTclf = tree.DecisionTreeClassifier()
DTclf = DTclf.fit(X_train, y_train) # Use X_train and y_train for fitting
prediction = DTclf.predict([[135, 29]])
print(prediction)
['Woman']
In [67]:
prediction = DTclf.predict_proba([[135,29]])
print(prediction)
[[0. 1.]]
In [ ]:
66°F
Smoke
Search
ENG
IN
11:49 PM
12/18/2023