

Linear Regression

```
# Import and create the model:
from sklearn.linear_model import LinearRegression

your_model = LinearRegression()
# Fit:
your_model.fit(x_training_data, y_training_data)
# .coef_: contains the coefficients
# .intercept_: contains the intercept
# Predict:
predictions = your_model.predict(your_x_data)
# .score(): returns the coefficient of determination R²
```

Naive Bayes

```
# Import and create the model:
from sklearn.naive_bayes import MultinomialNB

your_model = MultinomialNB()
# Fit:
your_model.fit(x_training_data, y_training_data)
# Predict:
# Returns a list of predicted classes - one prediction for every data point
predictions = your_model.predict(your_x_data)

# For every data point, returns a list of probabilities of each class
probabilities = your_model.predict_proba(your_x_data)
```

K-Nearest Neighbors

K-Means

```
# Import and create the model:
from sklearn.cluster import KMeans

your_model = KMeans(n_clusters=4, init='random')
n_clusters: number of clusters to form and number of centroids to generate
init: method for initialization
k-means++: K-Means++ [default]
random: K-Means
random_state: the seed used by the random number generator [optional]
# Fit:
your_model.fit(x_training_data)
# Predict:
predictions = your_model.predict(your_x_data)
```

Validating the Model

```
# Import and print accuracy, recall, precision, and F1 score:
from sklearn.metrics import accuracy_score, recall_score, precision_score, f1_score

print(accuracy_score(true_labels, guesses))
print(recall_score(true_labels, guesses))
print(precision_score(true_labels, guesses))
print(f1_score(true_labels, guesses))
# Import and print the confusion matrix:
from sklearn.metrics import confusion_matrix

print(confusion_matrix(true_labels, guesses))
```

Training Sets and Test Sets

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.8, test_size=0.2)
# train_size: the proportion of the dataset to include in the train split
# test_size: the proportion of the dataset to include in the test split
# random_state: the seed used by the random number generator [optional]
```

```
# Import and create the model:
from sklearn.neighbors import KNeighborsClassifier

your_model = KNeighborsClassifier()

# Fit:
your_model.fit(x_training_data, y_training_data)

# Predict:
# Returns a list of predicted classes - one
# prediction for every data point
predictions = your_model.predict(your_x_data)

# For every data point, returns a list of probabilities
# of each class
probabilities = your_model.predict_proba(your_x_data)
```



By **Ikashv**

cheatography.com/ikashv/

Published 23rd October, 2023.

Last updated 2nd January, 2023.

Page 1 of 2.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>