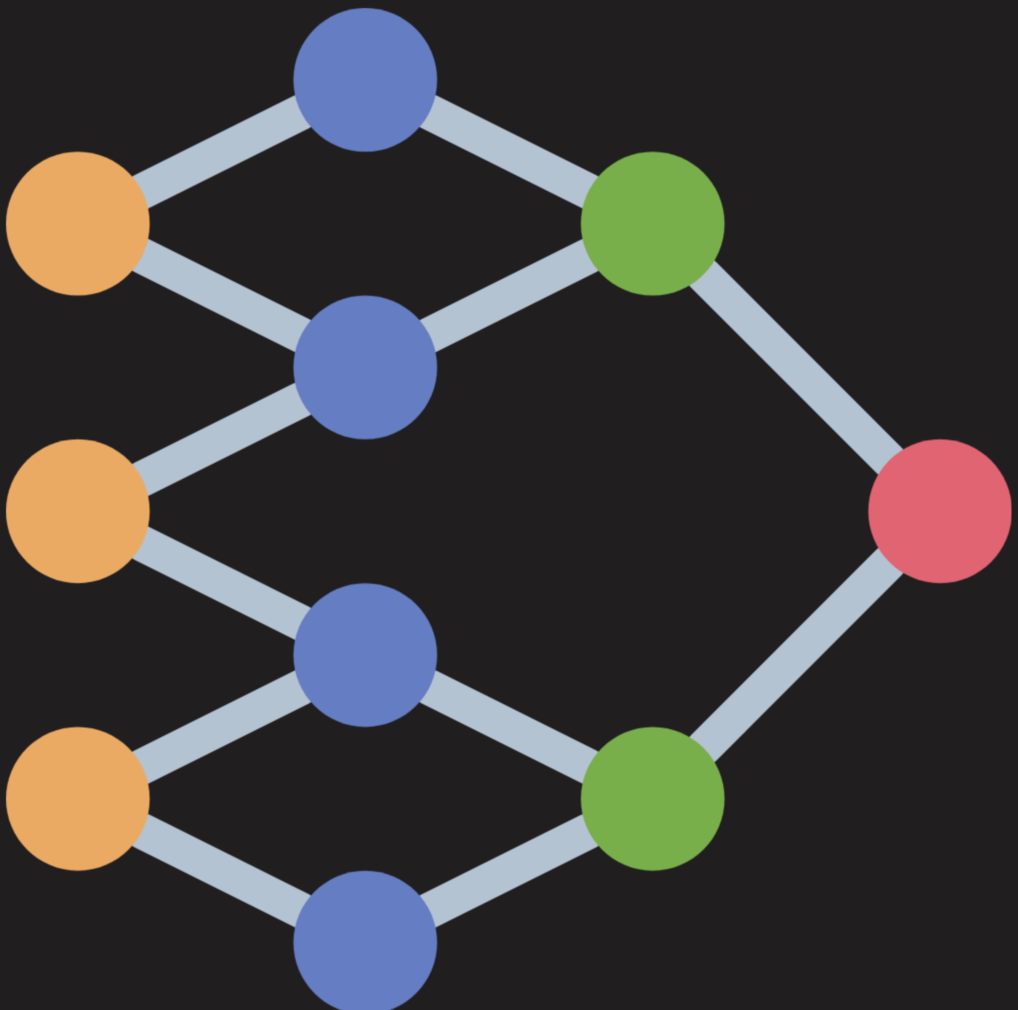
Python machine learning



By Sutton

Machine learning

An AI (Artificial Intelligence) model scientist specializes in developing, training, and evaluating machine learning models and algorithms that can perform specific tasks autonomously or semi-autonomously.

1. Supervised Learning and unsupervised Learning.

Supervised learning involves training a model on labeled data, where the target outcome is known. The model learns to map inputs to the desired output.

```
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

# Load data
data = load_iris()
X = data.data
y = data.target

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
```

Unsupervised learning deals with data without labeled responses. The goal is to identify patterns or structures within the data.

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Generate random data
X = [[1, 2], [1, 4], [1, 0], [10, 2], [10, 4], [10, 0]]

# K-means model
kmeans = KMeans(n_clusters=2, random_state=0).fit(X)

# Prediction and visualization
labels = kmeans.labels_
centroids = kmeans.cluster_centers_

plt.scatter([x[0] for x in X], [x[1] for x in X], c=labels)
plt.scatter(centroids[:, 0], centroids[:, 1], marker='x', s=100, c='red')
plt.show()
```

2. Data transformations. Data transformations are crucial to prepare the data for analysis, making it suitable for the machine learning algorithms.

```
from sklearn.preprocessing import StandardScaler

# Example data
data = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]

# Standardization
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)
print(scaled_data)
```

3. Regressors, classifiers and trees.

Regressors are used to predict continuous values.

```
from sklearn.linear_model import LinearRegression

# Example data
X = [[1], [2], [3], [4]]
y = [1.5, 3.0, 4.5, 6.0]

# Linear regression model
model = LinearRegression()
model.fit(X, y)

# Prediction
predictions = model.predict([[5]])
print(predictions)
```

Classifiers are used to predict discrete labels.

```
from sklearn.svm import SVC

# Example data
X = [[0, 0], [1, 1]]
y = [0, 1]

# SVM model
clf = SVC()
clf.fit(X, y)

# Prediction
predictions = clf.predict([[2, 2]])
print(predictions)
```

Decision trees use a tree-like model to make decisions based on features.

```
from sklearn.tree import DecisionTreeClassifier

# Example data
X = [[0, 0], [1, 1]]
y = [0, 1]

# Decision tree model
clf = DecisionTreeClassifier()
clf.fit(X, y)

# Prediction
predictions = clf.predict([[2, 2]])
print(predictions)
```

4. Feature selection models. Feature selection involves selecting the most relevant features for model training.

```
from sklearn.feature_selection import SelectKBest, f_classif

# Example data
X = [[0, 0, 1], [1, 1, 1], [0, 1, 0], [1, 0, 0]]
y = [0, 1, 0, 1]

# Feature selection
selector = SelectKBest(f_classif, k=2)
X_new = selector.fit_transform(X, y)
print(X_new)
```

5. Unsupervised learning algorithms. Unsupervised learning algorithms, such as Principal Component Analysis (PCA), reduce dimensionality and reveal hidden structures.

```
from sklearn.decomposition import PCA

# Example data
X = [[2.5, 2.4], [0.5, 0.7], [2.2, 2.9], [1.9, 2.2], [3.1, 3.0], [2.3, 2.7]]

# PCA
pca = PCA(n_components=1)
principalComponents = pca.fit_transform(X)
print(principalComponents)
```

6. Software engineering in Python. Software engineering in Python involves practices such as object-oriented programming and unit testing.

```python
class Calculator:
    def add(self, x, y):
        return x + y

    def subtract(self, x, y):
        return x - y

# Unit tests
import unittest

class TestCalculator(unittest.TestCase):
    def setUp(self):
        self.calc = Calculator()

    def test_add(self):
        self.assertEqual(self.calc.add(2, 3), 5)

    def test_subtract(self):
        self.assertEqual(self.calc.subtract(5, 3), 2)

if __name__ == '__main__':
    unittest.main()
```

7. Command line. Command line usage is essential for many development and data analysis tasks.

```bash
# List files in a directory
ls -l
```

8. Bash scripting. Bash scripting allows automation of tasks.

```bash
#!/bin/bash
# Script to create a directory and change permissions
mkdir my_directory
chmod 755 my_directory
```

9. Git and Python. Git is an essential tool for version control in software development.

```bash
# Initialize a Git repository
git init

# Add files and make a commit
git add .
git commit -m "Initial commit"
```

10. Supervised learning, advanced regressors and classifiers.

Advanced regressors like Ridge Regression handle regularization.

```
from sklearn.linear_model import Ridge

# Example data
X = [[0, 0], [0, 0], [1, 1]]
y = [0, .1, 1]

# Ridge model
clf = Ridge(alpha=1.0)
clf.fit(X, y)
print(clf.predict([[2, 2]]))
```

Advanced Classifiers like RandomForest can handle complex data.

```
from sklearn.ensemble import RandomForestClassifier

# Example data
X = [[0, 0], [1, 1]]
y = [0, 1]

# Random forest model
clf = RandomForestClassifier()
clf.fit(X, y)
print(clf.predict([[2, 2]]))
```

11. Deep learning. Deep learning is used for complex problems like image recognition.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Create a simple model
model = Sequential([
    Dense(10, activation='relu', input_shape=(4,)),
    Dense(3, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Example data
X = [[1, 2, 3, 4], [4, 5, 6, 7]]
y = [0, 1]

# Train the model
model.fit(X, y, epochs=10)
```

12. Natural language processing. Natural Language Processing (NLP) is used to work with text and linguistic data.

```
from sklearn.feature_extraction.text import CountVectorizer

# Example data
text = ["I love programming.", "Python is amazing."]

# Text vectorization
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(text)
print(X.toarray())
print(vectorizer.get_feature_names_out())
```

13. Regularization and hyperparameter turning. Regularization and hyperparameter tuning are essential for improving model performance.

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge

# Example data
X = [[0, 0], [0, 0], [1, 1]]
y = [0, .1, 1]

# Define parameters for GridSearch
parameters = {'alpha': [0.1, 1.0, 10.0]}
ridge = Ridge()

# GridSearchCV
clf = GridSearchCV(ridge, parameters)
clf.fit(X, y)
print(clf.best_params_)
```

14. Ensemble methods in machine learning. Ensemble methods combine multiple models to improve performance.

```
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

# Example data
X = [[0, 0], [1, 1]]
y = [0, 1]

# Base models
clf1 = LogisticRegression()
clf2 = SVC(probability=True)

# Ensemble model
eclf = VotingClassifier(estimators=[('lr', clf1), ('svc', clf2)], voting='soft')
eclf.fit(X, y)
print(eclf.predict([[2, 2]]))
```

15. Machine learning pipelines. Pipelines help structure and simplify the machine learning process.

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

# Example data
X = [[0, 0], [1, 1]]
y = [0, 1]

# Define pipeline
pipe = Pipeline([
    ('scaler', StandardScaler()),
    ('svc', SVC())
])

# Train model
pipe.fit(X, y)
print(pipe.predict([[2, 2]]))
```

16. Neural networks. Neural networks are fundamental in deep learning for complex tasks.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Create a neural network
model = Sequential([
    Dense(64, activation='relu', input_shape=(784,)),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Example data (MNIST)
mnist = tf.keras.datasets.mnist
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train, X_test = X_train / 255.0, X_test / 255.0

# Train the neural network
model.fit(X_train, y_train, epochs=5)
```