Machine Learning Models - Full Descriptions, Syntax, and Examples

Linear Regression

Description: Estimates relationship between variables using a straight line.
Example Use: Used for predicting continuous values like prices.
Syntax:

```
from sklearn.linear_model import LinearRegression model = LinearRegression()
model.fit(X_train, y_train) model.predict(X_test)
```

Ridge Regression

Description: Linear regression with L2 regularization to reduce overfitting.
Example Use: Good when features are correlated.
Syntax:

```
from sklearn.linear_model import Ridge model = Ridge(alpha=1.0) model.fit(X_train,
y_train)
```

Lasso Regression

Description: Linear regression with L1 regularization, can set coefficients to zero.
Example Use: Helps in feature selection.
Syntax:

```
from sklearn.linear_model import Lasso model = Lasso(alpha=0.1) model.fit(X_train,
y_train)
```

ElasticNet

Description: Combines L1 and L2 penalties for linear regression.
Example Use: Balances Ridge and Lasso benefits.
Syntax:

```
from sklearn.linear_model import ElasticNet model = ElasticNet(alpha=0.1,
l1_ratio=0.5) model.fit(X_train, y_train)
```

Polynomial Regression

Description: Models non-linear relationships using polynomial features.
Example Use: Extends linear regression to curve fitting.
Syntax:

```
from sklearn.preprocessing import PolynomialFeatures poly =
PolynomialFeatures(degree=2) X_poly = poly.fit_transform(X) model =
LinearRegression().fit(X_poly, y)
```

Logistic Regression

Description: Predicts probabilities for classification using the logistic function.
Example Use: Used for binary or multiclass classification.
Syntax:

```
from sklearn.linear_model import LogisticRegression model = LogisticRegression()
model.fit(X_train, y_train)
```

K-Nearest Neighbors (KNN)

Description: Classifies based on closest training samples.
Example Use: Simple and works well for small datasets.
Syntax:

```
from sklearn.neighbors import KNeighborsClassifier model =
KNeighborsClassifier(n_neighbors=5) model.fit(X_train, y_train)
```

Decision Tree Classifier

Description: Splits data into branches based on feature values.
Example Use: Easy to interpret and visualize.
Syntax:

```
from sklearn.tree import DecisionTreeClassifier model = DecisionTreeClassifier()
model.fit(X_train, y_train)
```

Random Forest Classifier

Description: Ensemble of decision trees using bagging.
Example Use: Handles overfitting better than a single tree.
Syntax:

```
from sklearn.ensemble import RandomForestClassifier model = RandomForestClassifier()
model.fit(X_train, y_train)
```

Gradient Boosting Classifier

Description: Builds trees sequentially to reduce errors.
Example Use: High accuracy, slower to train.
Syntax:

```
from sklearn.ensemble import GradientBoostingClassifier model =
GradientBoostingClassifier() model.fit(X_train, y_train)
```

Gaussian Naive Bayes

Description: Assumes features follow a Gaussian distribution.
Example Use: Works well with continuous data.
Syntax:

```
from sklearn.naive_bayes import GaussianNB model = GaussianNB() model.fit(X_train,
y_train)
```

Multinomial Naive Bayes

Description: Naive Bayes for count-based features.
Example Use: Common for text classification.
Syntax:

```
from sklearn.naive_bayes import MultinomialNB model = MultinomialNB()
model.fit(X_train, y_train)
```

Support Vector Classifier (SVC)

Description: Finds optimal hyperplane to separate classes.
Example Use: Effective in high-dimensional spaces.
Syntax:

```
from sklearn.svm import SVC model = SVC(kernel='rbf') model.fit(X_train, y_train)
```

Linear SVC

Description: Linear support vector machine.
Example Use: Faster than SVC for large datasets.

Syntax:

```
from sklearn.svm import LinearSVC model = LinearSVC() model.fit(X_train, y_train)
```

KMeans Clustering

Description: Partitions data into K clusters based on similarity.
Example Use: Unsupervised learning for grouping data.
Syntax:

```
from sklearn.cluster import KMeans model = KMeans(n_clusters=3) model.fit(X)
```

Agglomerative Clustering

Description: Hierarchical clustering that merges clusters iteratively.
Example Use: Good for hierarchical relationships.
Syntax:

```
from sklearn.cluster import AgglomerativeClustering model =
AgglomerativeClustering(n_clusters=3) model.fit(X)
```

Principal Component Analysis (PCA)

Description: Reduces dimensions by projecting onto principal components.
Example Use: Helps visualize high-dimensional data.
Syntax:

```
from sklearn.decomposition import PCA model = PCA(n_components=2) X_reduced =
model.fit_transform(X)
```

Truncated SVD

Description: Dimensionality reduction for sparse data.
Example Use: Used in NLP with sparse matrices.
Syntax:

```
from sklearn.decomposition import TruncatedSVD model = TruncatedSVD(n_components=2)
X_reduced = model.fit_transform(X)
```

Extra Trees Classifier

Description: Ensemble of randomized decision trees.
Example Use: Fast and reduces variance.
Syntax:

```
from sklearn.ensemble import ExtraTreesClassifier model = ExtraTreesClassifier()
model.fit(X_train, y_train)
```

Perceptron

Description: Simple linear binary classifier.
Example Use: Foundation of neural networks.
Syntax:

```
from sklearn.linear_model import Perceptron model = Perceptron() model.fit(X_train,
y_train)
```