



UNIVERSIDADE
CATÓLICA
PORTUGUESA

BRAGA

Machine Learning

Session 10 - PL

Linear Models

Degree in Applied Data Science

2024/2025

Linear Models in Scikit-Learn

- https://scikit-learn.org/stable/modules/linear_model.html

1.1. Linear Models

1.1.1. Ordinary Least Squares

1.1.2. Ridge regression and
classification

1.1.3. Lasso

1.1.4. Multi-task Lasso

1.1.5. Elastic-Net

1.1.6. Multi-task Elastic-Net

1.1.7. Least Angle Regression

1.1.8. LARS Lasso

1.1.9. Orthogonal Matching Pursuit
(OMP)

1.1.10. Bayesian Regression

1.1.11. Logistic regression

1.1.12. Generalized Linear Models

1.1.13. Stochastic Gradient

Descent - SGD

1.1.14. Perceptron

1.1.15. Passive Aggressive

Algorithms

1.1.16. Robustness regression:
outliers and modeling errors

1.1.17. Quantile Regression

1.1.18. Polynomial regression:
extending linear models with basis
functions

Linear Regression with Scikit-Learn

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html#sklearn-linear-model-linearregression

`sklearn.linear_model.LinearRegression`

```
class sklearn.linear_model.LinearRegression(*, fit_intercept=True, copy_X=True, n_jobs=None, positive=False)
```

[\[source\]](#)

```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
>>> X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
>>> # y = 1 * x_0 + 2 * x_1 + 3
>>> y = np.dot(X, np.array([1, 2])) + 3
>>> reg = LinearRegression().fit(X, y)
>>> reg.score(X, y)
1.0
>>> reg.coef_
array([1., 2.])
>>> reg.intercept_
3.0...
>>> reg.predict(np.array([[3, 5]]))
array([16.])
```

Ridge Regression with Scikit-Learn

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html#sklearn-linear-model-ridge

sklearn.linear_model.Ridge

```
class sklearn.linear_model.Ridge(alpha=1.0, *, fit_intercept=True, copy_X=True, max_iter=None, tol=0.0001,  
solver='auto', positive=False, random_state=None)
```

[\[source\]](#)

```
>>> from sklearn.linear_model import Ridge  
>>> import numpy as np  
>>> n_samples, n_features = 10, 5  
>>> rng = np.random.RandomState(0)  
>>> y = rng.randn(n_samples)  
>>> X = rng.randn(n_samples, n_features)  
>>> clf = Ridge(alpha=1.0)  
>>> clf.fit(X, y)  
Ridge()
```

Lasso Regression with Scikit-Learn

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html#sklearn-linear-model-lasso

`sklearn.linear_model.Lasso`

```
class sklearn.linear_model.Lasso(alpha=1.0, *, fit_intercept=True, precompute=False, copy_X=True, max_iter=1000,  
tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic')
```

[\[source\]](#)

```
>>> from sklearn import linear_model  
>>> clf = linear_model.Lasso(alpha=0.1)  
>>> clf.fit([[0,0], [1, 1], [2, 2]], [0, 1, 2])  
Lasso(alpha=0.1)  
>>> print(clf.coef_)  
[0.85 0. ]  
>>> print(clf.intercept_)  
0.15...
```

Logistic Regression with Scikit-Learn

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn-linear-model-logisticregression

`sklearn.linear_model.LogisticRegression`

```
class sklearn.linear_model.LogisticRegression(penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True,
intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0,
warm_start=False, n_jobs=None, l1_ratio=None)
```

[\[source\]](#)

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.linear_model import LogisticRegression
>>> X, y = load_iris(return_X_y=True)
>>> clf = LogisticRegression(random_state=0).fit(X, y)
>>> clf.predict(X[:2, :])
array([0, 0])
>>> clf.predict_proba(X[:2, :])
array([[9.8...e-01, 1.8...e-02, 1.4...e-08],
       [9.7...e-01, 2.8...e-02, ...e-08]])
>>> clf.score(X, y)
0.97...
```

Exercises:

- Notebooks on the github repository:
 - Notebook with examples:
 - `exercises/session10/examples.ipynb`
 - Notebook with exercises:
 - `exercises/session10/exercises.ipynb`