

Cheat Sheet: Linear and Logistic Regression

Comparing different regression types

| Model Name | Description | Code Syntax |
|----------------------------|---|--|
| Simple linear regression | <p>Purpose: To predict a dependent variable based on one independent variable.</p> <p>Pros: Easy to implement, interpret, and efficient for small datasets.</p> <p>Cons: Not suitable for complex relationships; prone to underfitting.</p> <p>Modeling equation: $y = b_0 + b_1x$</p> | <pre>from sklearn.linear_model import LinearRegression model = LinearRegression() model.fit(X, y)</pre> |
| Polynomial regression | <p>Purpose: To capture nonlinear relationships between variables.</p> <p>Pros: Better at fitting nonlinear data compared to linear regression.</p> <p>Cons: Prone to overfitting with high-degree polynomials.</p> <p>Modeling equation: $y = b_0 + b_1x + b_2x^2 + \dots$</p> | <pre>from sklearn.preprocessing import PolynomialFeatures from sklearn.linear_model import LinearRegression poly = PolynomialFeatures(degree=2) X_poly = poly.fit_transform(X) model = LinearRegression().fit(X_poly, y)</pre> |
| Multiple linear regression | <p>Purpose: To predict a dependent variable based on multiple independent variables.</p> <p>Pros: Accounts for multiple factors influencing the outcome.</p> <p>Cons: Assumes a linear relationship between predictors and target.</p> <p>Modeling equation: $y = b_0 + b_1x_1 + b_2x_2 + \dots$</p> | <pre>from sklearn.linear_model import LinearRegression model = LinearRegression() model.fit(X, y)</pre> |
| Logistic regression | <p>Purpose: To predict probabilities of categorical outcomes.</p> <p>Pros: Efficient for binary classification problems.</p> <p>Cons: Assumes a linear relationship between independent variables and log-odds.</p> <p>Modeling equation: $\log(p/(1-p)) = b_0 + b_1x_1 + \dots$</p> | <pre>from sklearn.linear_model import LogisticRegression model = LogisticRegression() model.fit(X, y)</pre> |

Associated functions commonly used

| Function/Method Name | Brief Description | Code Syntax |
|----------------------|---|---|
| train_test_split | Splits the dataset into training and testing subsets to evaluate the model's performance. | <pre>from sklearn.model_selection import train_test_split X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)</pre> |
| StandardScaler | Standardizes features by removing the mean and scaling to unit variance. | <pre>from sklearn.preprocessing import StandardScaler scaler = StandardScaler() X_scaled = scaler.fit_transform(X)</pre> |
| log_loss | Calculates the logarithmic loss, a performance metric for classification models. | <pre>from sklearn.metrics import log_loss loss = log_loss(y_true, y_pred_proba)</pre> |
| mean_absolute_error | Calculates the mean absolute error between actual and predicted values. | <pre>from sklearn.metrics import mean_absolute_error mae = mean_absolute_error(y_true, y_pred)</pre> |
| mean_squared_error | Computes the mean squared error between actual and predicted values. | <pre>from sklearn.metrics import mean_squared_error mse = mean_squared_error(y_true, y_pred)</pre> |

| Function/Method Name | Brief Description | Code Syntax |
|-------------------------|--|---|
| root_mean_squared_error | Calculates the root mean squared error (RMSE), a commonly used metric for regression tasks. | <pre>from sklearn.metrics import mean_squared_error import numpy as np rmse = np.sqrt(mean_squared_error(y_true, y_pred))</pre> |
| r2_score | Computes the R-squared value, indicating how well the model explains the variability of the target variable. | <pre>from sklearn.metrics import r2_score r2 = r2_score(y_true, y_pred)</pre> |

Author(s)

[Jeff Grossman](#)
[Abhishek Gagneja](#)



Skills Network