# Cheat Sheet: Linear and Logistic Regression

## Comparing different regression types

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

## 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. | `from sklearn.model_selection import train_test_split`<br>`X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)` |
| StandardScaler | Standardizes features by removing the mean and scaling to unit variance. | `from sklearn.preprocessing import StandardScaler`<br>`scaler = StandardScaler()`<br>`X_scaled = scaler.fit_transform(X)` |

| Function/Method Name | Brief Description | Code Syntax |
|---|---|---|
| | | |
| log_loss | Calculates the logarithmic loss, a performance metric for classification models. | `from sklearn.metrics import log_loss`<br>`loss = log_loss(y_true, y_pred_proba)` |
| mean_absolute_error | Calculates the mean absolute error between actual and predicted values. | `from sklearn.metrics import mean_absolute_error`<br>`mae = mean_absolute_error(y_true, y_pred)` |
| mean_squared_error | Computes the mean squared error between actual and predicted values. | `from sklearn.metrics import mean_squared_error`<br>`mse = mean_squared_error(y_true, y_pred)` |
| root_mean_squared_error | Calculates the root mean squared error (RMSE), a commonly used metric for regression tasks. | `from sklearn.metrics import mean_squared_error`<br>`import numpy as np`<br>`rmse = np.sqrt(mean_squared_error(y_true, y_pred))` |
| r2_score | Computes the R-squared value, indicating how well the model explains the variability of the target variable. | `from sklearn.metrics import r2_score`<br>`r2 = r2_score(y_true, y_pred)` |

## Author(s)

Jeff Grossman
Abhishek Gagneja