**MODEL DEVELOPMENT**

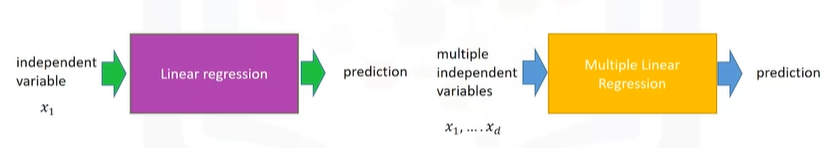
Learning objectives:

* Simple and multiple linear regression
* Model evaluation using visualization
* Polynomial regression and pipelines
* R-squared and MSE for in-sample evaluation
* Prediction and decision making
* A model can be thought of as a mathematical equation used to predict a value given one or more other values
* Relating one or more independent variables to dependent variables
* Usually, the more relevant data you have the more accurate your model is.

**LINEAR REGRESSION AND MULTIPLE LINEAR REGRESSION**

Linear regression will refer to 1 independent variable to make a prediction

Multiple linear regression will refer to multiple independent variables to make a prediction



**SIMPLE LINEAR REGRESSION (SLR)**

1. The predictor (independent) variable – x
2. The target (dependent) variable – y

With: b0: the intercept and b1: the slope

**FITTING A SLR ESTIMATOR:**

X: predictor variable

Y: target variable

1. Import linear\_model from scikit-learn

From sklearn.linear\_model import LinearRegression

1. Create a linear regression object using the constructor

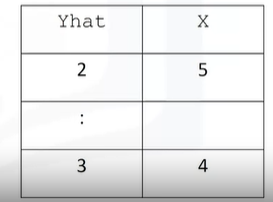
lm=LinearRegression()

We define the predictor variable and target variable

X = df[[‘highway-mpg’]]

Y = df[‘price’]

Then use lm.fit(X, Y) to fit the model, i.e fine the parameter b0 and b1

lm.fit(X, Y)

We obtain a prediction

Yhat = lm.predict(X)

**Text

Description automatically generatedMULTIPLE LINEAR REGRESSION (MLR)**

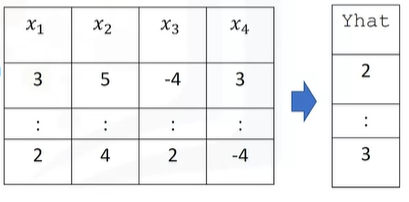
Chart

Description automatically generated

**Fitting a multiple linear regression (MLR)**

1. We can extract the 4 predictor variables and store them in the variable Z

Z = df[[‘horse-power’, ‘curb-weight’, ‘engine-size’, ‘highway-mpg’]]

1. Then train the model as before

lm.fit(Z, df[‘price’])

1. We can also obtain a prediction

Yhat = lm.predict(X)

Find the intercept (b0): lm.intercept\_

Find the coefficients (b1, b2, b3, b4): lm.coef\_

**Chart, scatter chart

Description automatically generatedREGRESSION PLOT:**

Regression plot shows us a combination of:

The scatterplot: where each point represents a different Y

The fitted linear regression line (Yhat)

Import seaborn as sns

Sns.regplot(x = “highway-mpg”, y = “price”, data= df)

Plt.ylim(0,)

**A picture containing chart

Description automatically generated RESIDUAL PLOT**

Chart, scatter chart

Description automatically generated

Chart, line chart

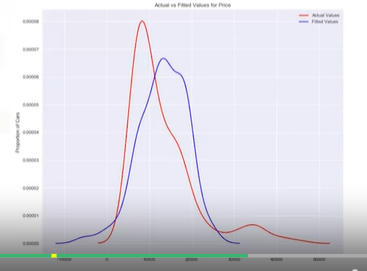
Description automatically generated

Chart, scatter chart

Description automatically generated

Import seaborn as sns

Sns.residplot(df[‘highway-mpg’], df[‘price’])



**DISTRIBUTION PLOTS**

Compare the distribution plots:

The fitted values that result from the model

The actual values

Import seaborn as sns

Ax1 = sns.displot(df[‘price’], hist= False, color=”r”, label= “Actual Value”)

Sns.displot(Yhat, hist= False, color= “b”, label= “Fitted Value”, ax=ax1)

Chart, line chart, scatter chart

Description automatically generated

**POLYNOMIAL REGRESSION AND PIPELINES**

**POLYNOMIAL REGRESSION**

A special case of the general linear regression model

Useful for describing curvilinear relationships

Curvilinear relationships:

By squaring or setting higher-order terms of the predictor vartiables

Diagram, text

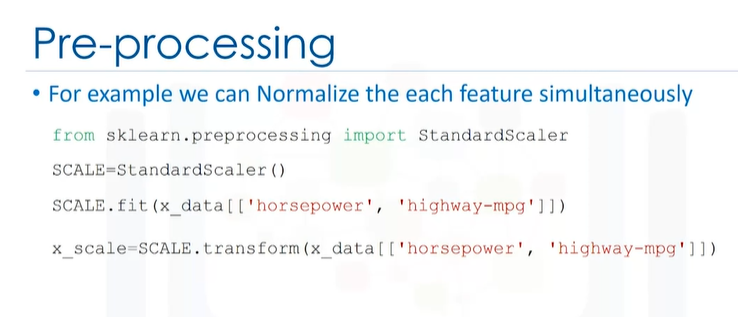
Description automatically generated with medium confidence

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, table

Description automatically generated



**PIPELINES:**

Diagram

Description automatically generated

From sklearn.preprocessing import PolynomialFeatures

From sklearn.linear\_model import LinearRegression

From sklearn.preprocessing import StandardScaler

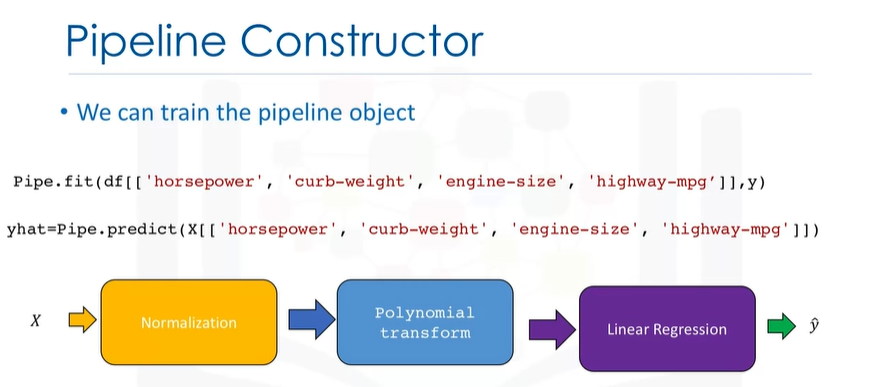
From sklearn.pipeline import Pipeline

A picture containing timeline

Description automatically generated

Blue: name of the estimator – model

Red: model constructor



**MEASURE FOR IN-SAMPLE EVALUATION**

* A way to numerically determine how good the model fits on dataset
* Two important measures to determine the fit of a model
* Mean squared error (MSE)
* R-squared (R2)

**Chart, scatter chart

Description automatically generatedMean squared error (MSE)**

In python, we can measure the MSE as follows

From sklearn.metrics import mean\_squared\_error

Mean\_squared\_error(df[‘price’], Y\_predict\_simple\_fit)

**R-squared (R2)**

The coefficient of determination or R-squared (R2)

Is a masure to determine how close the data is to the fitted regression line

R2: the percentage of variation of the target variable (Y) that is explained by the linear model

A picture containing diagram

Description automatically generatedThink about as comparing a regression model to a simple model i.e the mean of the data points

Generally, the values of the MSE are between 0 and 1.

We can calculate the R2 as follows:

X = df[[‘highway-mog’]]

Y = df[‘price’]

Lm.fit(X, Y)

Lm.score(X, Y)

**PREDICTION AND DECISION MAKING**

**DECISION MAKING: DETERMINING A GOOD MODEL FIT**

To determine final best fit, we look at a combination of:

* Do the predicted values make sense
* Visualization
* Numerical measures for evaluation
* Comparing models

**Comparing MLR and SLR**

MSE for MLR model will be smaller than the MSE for a SLR model, since the errors of the data will decrease when more variables are included in the model

Polynomial regression will also have a smaller MSE than the linear regular regression

Now that we have visualized the different models, and generated the R-squared and MSE values for the fits, how do we determine a good model fit?

* What is a good R-squared value?

**When comparing models, the model with the higher R-squared value is a better fit for the data.**

* What is a good MSE?

**When comparing models, the model with the smallest MSE value is a better fit for the data.**