Python Mod 5 Parte 2

= Model Development =

- 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 independant variables to dependant variables

Independant variables or features ⇒ Model ⇒ dependant variables

Km por litro ⇒ Model ⇒ Predicted Price

- Usually the more relevant data you have the more accurate your model is (Independant variables)

Examples of models:
- Simple Linear Regression
- ✓ Multiple Linear Regression
- ✓ Polynomial ~~Linear~~ Regression

= SIMPLE LINEAR & MULTIPLE LINEAR REGRESSION =

- Linear regression will refer to one independant variable to make a prediction
- Multiple Linear Regresseon will refer to multiple independant variables to make a prediction.

Independent variable X_1 ⇒ Linear Regression ⇒ Prediction

Multiple Independant $X_1, \dots X_3$ Variables or Features ⇒ Multiple Regression ⇒ Prediction

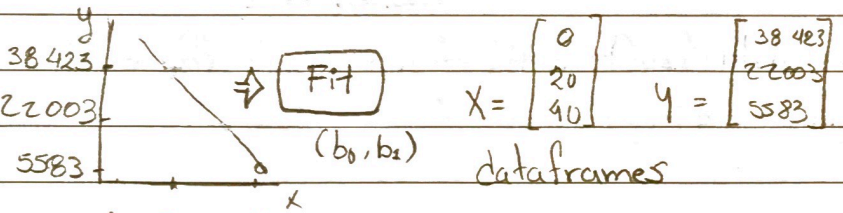
Simple Linear Regression

1.- The predictor (independent) variable X
2.- The target (dependent) variable y

$$y = b_0 + b_1 X$$

b_0 : the intercept
b_1 : the slope (pendiente)

1. Come up with a model



$X = \begin{bmatrix} 0 \\ 20 \\ 40 \end{bmatrix}$ $y = \begin{bmatrix} 38\,423 \\ 22003 \\ 5583 \end{bmatrix}$ dataframes

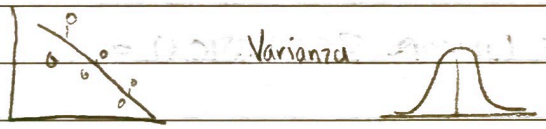
2.- Use training points to fit our model

3.- The result of the training points are the parameters

4.- Data is stored in dataframes

$\hat{y} = b_0 + b_1 X$

Usamos y gorrito para denotar que el modelo es un estimado



Varianza

Que tan lejos los puntos están de la línea (mi modelo), el error.

: Fitting a Simple Linear Model Estimator :

1.- Import linear_model from scikit-learn
```
from sklearn.linear_model import LinearRegression
```
2.- Create a Linear Regression Object using the constructor:
```
lm = LinearRegression()
```
3.- We define the predictor value and target ~~value~~ variable
```
X = df[['highway-mpg']]
y = df['price']
```
4.- Then use `lm.fit(x,y)` to fit the model, find the parameters b_0 and b_1
```
lm.fit(x,y)
```

Norma

5.- We can obtain a prediction

$\hat{y} = lm.predict(x)$

Multiple Linear Regression (MLR)

This method is used to explain the relationship between:

1. One continuous target (Y) variable
2. Two or more predictor (x) variable

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4$$

b_0 = intercept (x=0)

b_1 = the coefficient or parameter of x_1

b_2 = the coefficient or parameter of x_2 and so on

Fitting a Multiple Linear Model Estimator

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

```
z = df[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']]
```

2. Train the model as before

```
lm.fit(z, df['price'])
```

3. We can also obtain a prediction

```
ŷ = lm.predict(x)
```

= Model Evaluation using Visualization =

Regression Plot

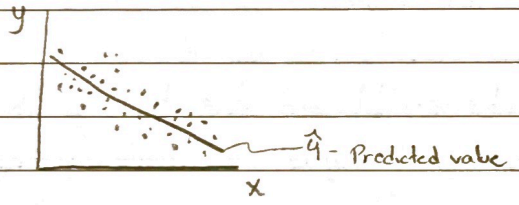
* Why we regression plot?

--→ It gives us a good estimate of:

1- The relationship between two variables
2- The strenght of the correlation
3- The direction of the relationship (positive or negative)

Regression Plot shows us a combination of:

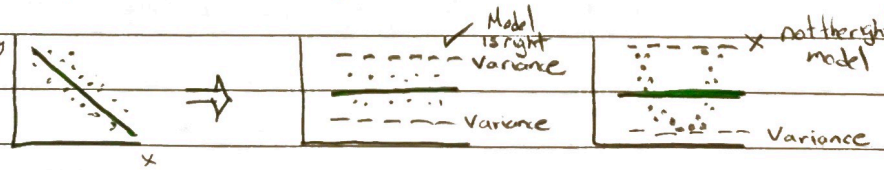
- The scatterplot: where each point represents a different y
- The fitted linear regression line ($\hat{y}$)
- Horizontal axis is the independent variable (x)
- Vertical axis is the dependent variable (y)




```
import seaborn as sns
sns.regplot(x="highway-mpg", y="price", data=df)
plt.ylim(0,)
```

Residual Plot

It is a graph that is used to examine the goodness-of-fit in regression
→ Muestra que tanto error hay en el modelo (fitted line-curve)

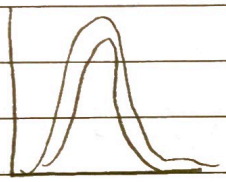


Residual Plot

```
import seaborn as sns
sns.residplot(df['highway-mpg'], df['price'])
```

= Distribution Plot =

Distribution Plots are for continuous (float → temperature in °C) values
Histograms are for discrete (int → age, No. of students) values

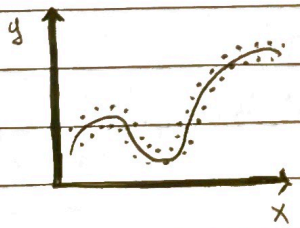


```
import seaborn as sns
ax1 = sns.distplot(df['price'], hist=False, color="r", label="Actual Value")
sns.distplot(Yhat, hist=False, color="b", label="Fitted Values", ax=ax1)
```

Polynomial Regression ~~with more than one dimension~~
= and =
Pipelines

- A special case of the general linear regression model
- Useful for describing curvilinear relationships

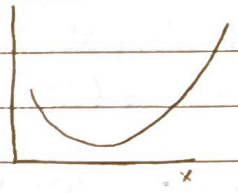
Curvilinear relationships:
By squaring or setting higher-order terms of the predictor variables



Polynomial Regression

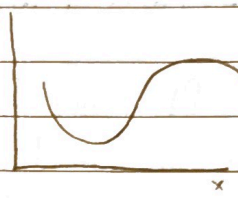
- Quadratic - 2nd order

$$\hat{y} = b_0 + b_1 x_1 + b_2 (x_1)^2$$



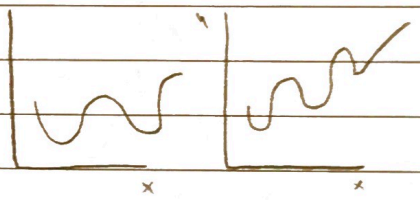
- Cubic - 3rd order

$$\hat{y} = b_0 + b_1 x_1 + b_2 (x_1)^2 + b_3 (x_1)^3$$



- Higher order

$$\hat{y} = b_0 + b_1 x_1 + b_2 (x_1)^2 + b_3 (x_1)^3 + \dots$$



The degree of the regression makes a big difference and can result in a better fit

- The expression can get complicated ei some terms for a two-dimensional second order polynomial:

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_1 x_2 + b_4 (x_1)^2 + b_5 (x_2)^2 + \dots$$

Polynomial Regression with More than One dimension

→ Pre-processing: Normalize each feature (independent variables) simultaneously:

```
from sklearn.preprocessing import Standar Scaler
SCALE = StahdardScaler()
SCALE.fit(x_data[['horsepower', 'hyway-mpg']])
x_scale = SCALE.transform(x_data[['horsepower', 'hyhaympg']])
```

Pipelines

We can simplify our code by using a pipeline. Pipelines sequentially perform a series of transformations. The last step carries out a prediction

Normalization → Polynomial Transform → Linear Regression

Transformations | Prediction

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

```
Input = [('scale', StandardScaler()), ('polynomial', PolynomialFeatures
(degree=2)), ... ('model', LinearRegression())]
```

This is a list of tuples.

Contains the name of the estimator
The second element contain model constructor.

- Pipeline constructor

```
pipe = Pipeline(Input)
```

Pipeline object

```
Pipe.train(x
ŷ = Pipe.predict
```

↳ Ejecuta primero los pasos que indicamos arriba.

Measures 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
  - R-squared (R^2)

Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

```
from sklearn.metrics import mean_squared_error
mean_squared_error(df['price'], Y_predict_simple_fit)
```

Promedio del cuadrado de los errores

The function gets two inputs: the actual value of target variable and the predicted value of target variable.

R-squared / R^2

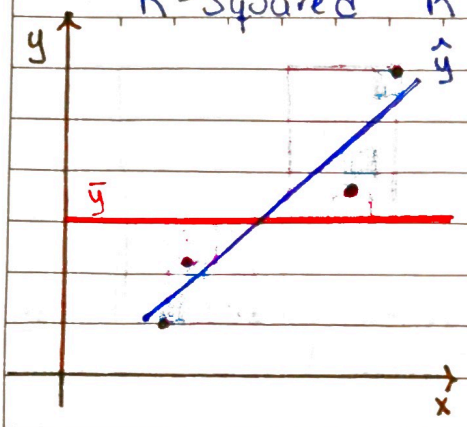
- The Coeffient of Determination or R squared (R^2)
- Is a measure to determine how close the data is to the fitted regression line.
- R^2: the percentage of variation of the target variable (

$$R^2 = 1 - \frac{\text{Sum of squared regression}}{\text{total sum of squares}}$$

$$= 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

$$= 1 - \frac{\text{MSE of regression line}}{\text{MSE of the average of the data}}$$

R-squared R^2



Regression line
Mean Squared Error of the regression line
Average value of the data points
Mean Squared error of the red line

We see blue squares is much smaller than the area of the red squares ∴ performs well

Close to 1 - ok
Close to 0 - not ok

Generally the values of the MSE are between 0 and 1
We can calculate the R^2 as follows

```
X = df[['highway-mpg']]
Y = df['price']
lm.fit(X, Y)
lm.score(X, Y)
    0.4965
```

R^2 is usually between 0 and 1
if R^2 is negative it can be due to overfitting

PREDICTION AND 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
- Compare vs other model

Comparing MLR and SLR

1. Is a lower MSE always implying a better fit?
   - Not necessarily
2. MSE for an MLR model will be smaller than the MSE for an SLR model, since the errors of the data will ~~increase~~ decrease when more variables are included in the model
3. Polynomial regression will also have a smaller MSE than regular regression
4. A similar inverse relationship hold R^2