Introduction to Data Science with Python
Week 0, Section 1: Linear Regression

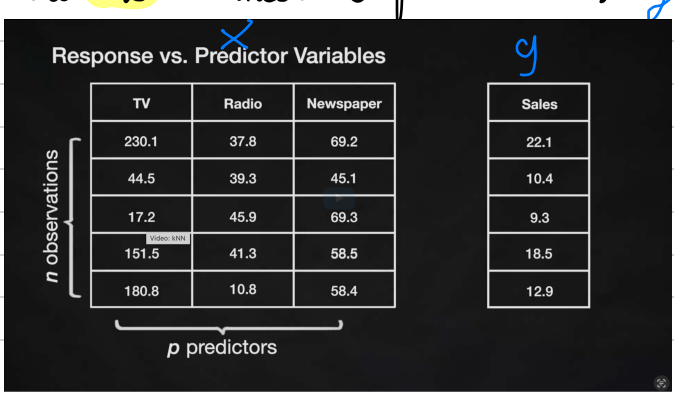
Book: An introduction to Statistical Learning, 2nd

Our main goal in data science is to predict outcomes and analyze data.

When dealing with this kinds of problems there is a common asymmetry: the variable we want to predict may be difficult, more important than others, or might be influenced by others. Which leaves us with 2 types of variables:

1 values we want to predict. (X)

2 values we use to make a prediction. (y)



Capital Letters = matrix

lower case = vectors

Pandas Correction:

- X.shape = returns dimensions in (n, p) format
- Y.shape = returns (n,) or (n, 1)
- df[['x']] vs df['x']: double returns a pd.dataframe, and single returns pd.series.

Statistical Model:

Assume that variable Y relates to predictor/s X, through an unknown $F(X)$ function and a random amount ε, that describes Y from the rule $f(x)$

$$Y = F(X) + \varepsilon$$

When we use a set of measurements $(x_{i1} \dots x_{ip})$ to predict a value we use

$$\hat{y} = \hat{F}(x_{i1}, \dots, x_{ip})$$

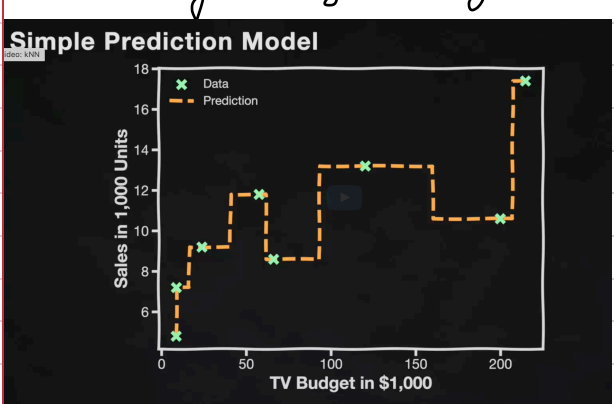
when we care about F's form. We call it inference problems. For some problems we don't care about the specific F form, these are prediction problems

We can take the mean of all y's with $\frac{1}{n}\sum_{i=1}^{n} y_i$

In our specific dataset we can find the distances to all other points $D(x_q, x_i)$. We are looking to answer the sales for TV budget of 175,000k.

We find the nearest neighbor (x_p, y_p) and predict $\hat{y}_q = y_q$

Prediction for any budget:



1.2 Error evaluation and model comparison

first we're gonna learn how to determine how to choose models $K=1$, $K=10$, $K=70$. To determine this we need to calculate the error each model does.

Error Evaluation:

- first, we randomly hide some data from the model. This is called train-validation, that is done to check how the model does with unseen data.
- we use the train set to estimate $\hat{y}$, and the validation set to evaluate the model.
- we bring back the validation data, an error is the difference between the prediction and the true value. $r_i = |y_i - \hat{y}_i|$
- then to quantify how well the model performs,

we aggregate the errors, and that's called loss, error, or cost function

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

Model Comparison

- After getting all MSE's, we choose the one closer to 0.

Model fitness

- To know if your model is good enough, we compare to a really good and bad models. In this case we have $\hat{y} = \bar{y}$ $1/n \sum_i y_i$ for the worst, and $\hat{y}_i = y_i$ as the best one. Were using $K = 3$, $MSE = s$.
- then we create the entity R-squared

$$R\text{-squared} = 1 - \frac{\sum_i (\hat{y}_i - y_i)^2}{\sum_i (\bar{y}_i - y_i)^2}$$

if the model gets a 1, the model is perfect, and a 0

if it is as good as the mean value $\bar{y}$, if it is a negative value then it's worse than average.

1.3 Linear Regression

What happens when now we ask "How will sales go if we double the TV budget?"

Linear Models

To build one, we must a simple form of f

$$f(x) = \beta_0 + \beta_1 X$$

And then we add the estimate

$$\hat{y} = \hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 X$$

$\hat{\beta}_1$ and $\hat{\beta}_0$ are estimates of β_1 and β_0

How to choose a line

We can use MSE as our loss function

$$L(\beta_0, \beta_1) = \frac{1}{n} \sum_{i=1}^{n} [y_i - (\beta_i x_i + \beta_0)]^2$$

The optimal values of the betas should be the minimum of the loss functions with respect to beta 0 and 1. To minimize loss.

Video: Linear Regression

Recap Exercise

```
>>> from sklearn.linear_model import LinearRegression
>>> df = pd.read_csv('Advertising.csv')
>>> X = df[['TV']].values
>>> y = df['Sales'].values
>>> reg = LinearRegression()
>>> reg.fit(X, y)
>>> reg.coef_
array([[0.04665056]])
>>> reg.intercept_
array([7.08543108])
>>> reg.predict(np.array([[100]]))
array([[11.75048733]])
```

HX

reg.fit(X, y)

chain rule = if a function depends on another function that depends on a variable.

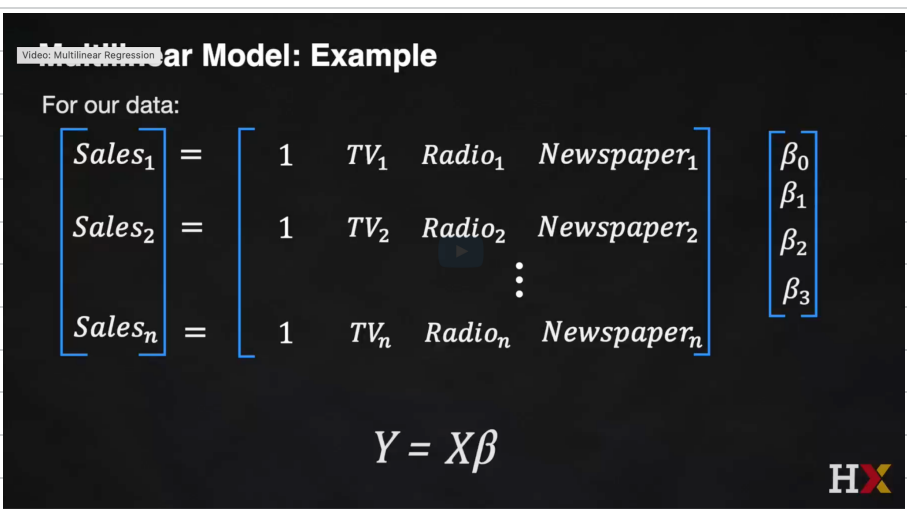
Section 2: Multiple and Polynomial Regression

2.1 Multiple Regression

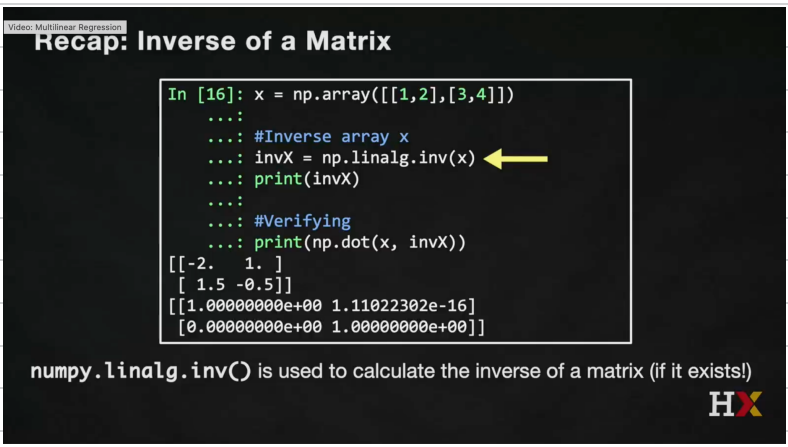
The goal now is for our model to take as much data as possible with multiple variables. When you add one more predictor, we have a 3D model

$$Sales_n = \beta_0 + \beta_1 * TV_n + \beta_2 * Radio_n + \beta_3 * Newspaper_n$$

$$Sales_1 = \beta_0 + [TV_1 * Radio_1 * newspaper_1) \cdot \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}$$



We can transpose a Matrix, which is moving it from vertical to horizontal or vicevirsa, using Numpy's `np.transpose()`



When doing multiple linear regression, we still use MSE as a loss function to keep $Y = X\beta$, then we take the partial derivative and equal it to 0

$$X^T(Y - X\beta) = 0$$

$$X^T X\beta = X^T Y$$

$$(X^T X)^{-1} X^T X\beta = (X^T X)^{-1} X^T Y$$

$$\beta = (X^T X)^{-1} X^T y$$

Practice:

We used Advertising.csv to predict sales based on 3 predictors

Linear Models and R^2:

we used fit_and_plot_linear, which returns graphs, then we used fit_and_plot_multi to use multiple variables, this returned R^2 values.

Predictions:

Learned how to use itertools.combinations to get all possible combinations of predictors and then append to a list, this uses a for loop in a range. Then to start evaluating losses we set `x = df[i]` and `y = df['Sales']`, then using train_test_split to assign train size to 80%, assign LinearRegression to a variable

Predict using .predict(x_test) and you can then get the MSE using mean_square_error

2.2 Techniques for Multilinear Modeling

Qualitative Predictors

Video: Multilinear Techniques

Qualitative Predictors

Income	Limit	Rating	Cards	Age	Education	Sex	Student	Married	Ethnicity	Balance
14.890	3606	283	2	34	11	Male	No	Yes	Caucasian	333
106.02	6645	483	3	82	15	Female	Yes	Yes	Asian	903
104.59	7075	514	4	71	11	Male	No	No	Asian	580
148.92	9504	681	3	36	11	Female	No	No	African American	964
55.882	4897	357	2	68	16	Male	No	Yes	Caucasian	331

quantitative qualitative

HX

If the predictor only takes two variables, we create a dummy variable. For sex we can equal 1 to male and 0 to female.

β_0 = males, $\beta_0 + \beta_1$ = females, β_i = difference between males and females. We create dumy variables for ethnicity we need an extra one. The idea is to find correlations.

Scaling

In an issue where we need a parabola but our code only gives straight lines, we need to scale the predictor, that fixes this situation but why? Generally, it works well when the predictors are not in the right magnitude, but it can also affect because of how sklearn solves a regression, when inverting a matrix it can fail, for which scaling can be helpful.

Normalize = Linear scaling into 0-1 range

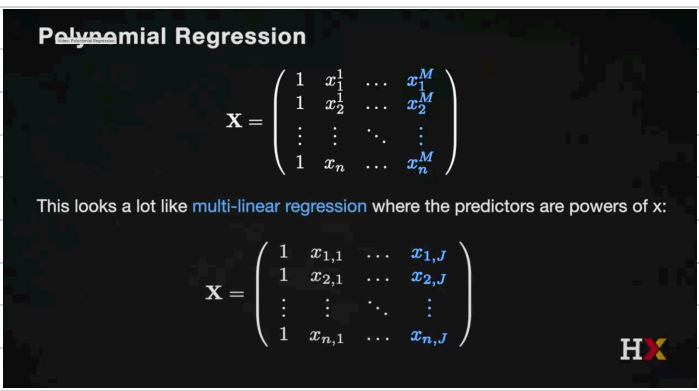
Standardize = Gaussian scaling around origin

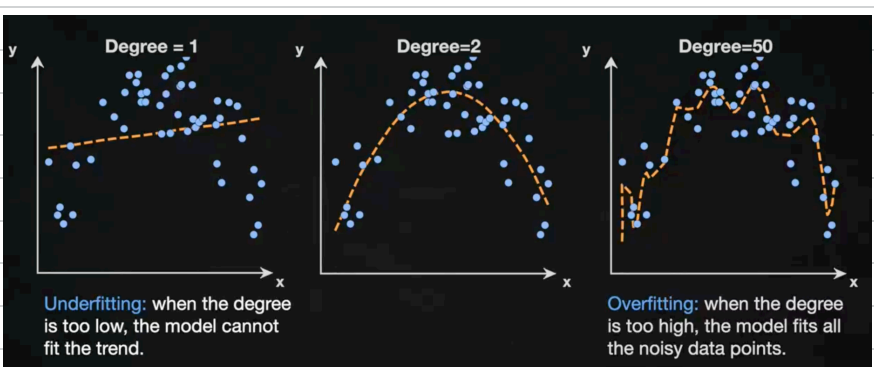
Practice

We learned about categories and how to set strings to booleans by using dummy's, then you include them in R^2's

2.3 Polynomial Regression

Sometimes our models might not be linear. Basically, $y = F_\beta(x)$ where F is not a linear function and β is a parameter of F

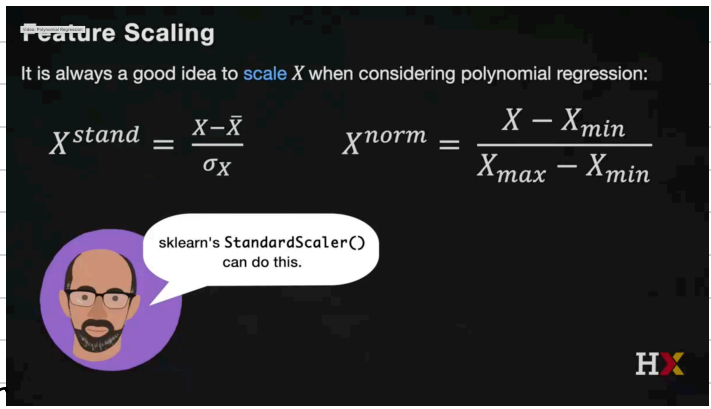




Feature Scaling

normally when the range of x is small or large

can cause some problems that can be solved by scalation



↳ This are the ways to scale a matrix