

# Week 12:

## Python tutorial

- Exploratory Data Analysis
- Simple linear Regression with CI
- Multiple regression analysis
- Dealing with missing data

# Multiple Regression Model

A general additive multiple regression model, which relates a dependent variable  $y$  to  $k$  predictor variables  $x_1, x_2, \dots, x_k$ , is given by the model equation

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k + e$$

The random deviation  $e$  is assumed to be normally distributed with mean value 0 and standard deviation  $s$  for any particular values  $x_1, \dots, x_k$ .

# Qualitative Predictors

Dealing with predictors that are **qualitative**.

**Example:** The Credit data set contains information about balance, age, cards, education, income, limit, and rating for a number of potential customers.

Income	Limit	Rating	Cards	Age	Education	Gender	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	Asian	964
55.882	4897	357	2	68	16	Male	No	Yes	Caucasian	331

# Qualitative Predictors

If the predictor takes only two values, then we create an **indicator** or **dummy variable** that takes on two possible numerical values.

For example for the gender, we create a new variable:

$$x_i = \begin{cases} 1 & \text{if } i \text{ th person is female} \\ 0 & \text{if } i \text{ th person is male} \end{cases}$$

We then use this variable as a predictor in the regression equation.

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i \text{ th person is female} \\ \beta_0 + \epsilon_i & \text{if } i \text{ th person is male} \end{cases}$$

# Dealing with interaction term

For example, if you have two independent variables  $X_1$  and  $X_2$ , and you suspect that the effect of  $X_1$  on the dependent variable  $Y$  depends on the level of  $X_2$ , you will include an interaction term  $X_1 \times X_2$  in your model:

**We change**

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

**To**

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \boxed{\beta_3 X_1 X_2} + \epsilon$$

Captures the interaction effect between  $X_1$  and  $X_2$

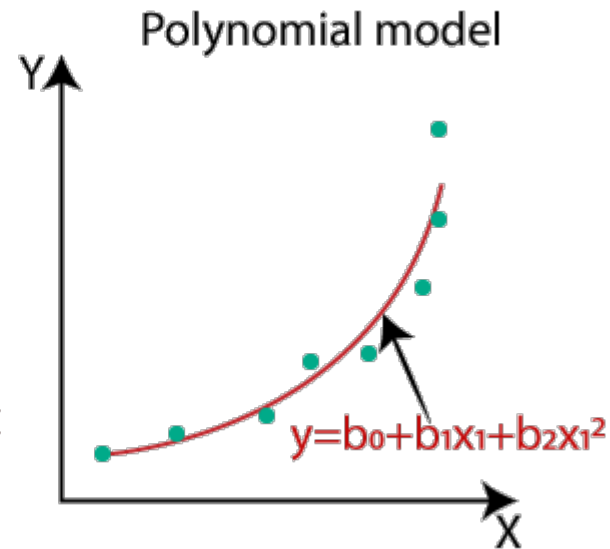
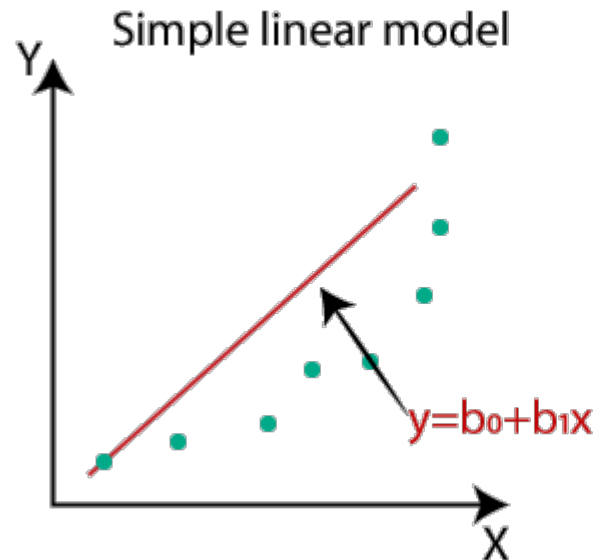
# Polynomial Regression

The  $n^{\text{th}}$  degree polynomial regression model is

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \dots + \beta_n x^n + \epsilon$$

is a special case of the general multiple regression model with

$$x_1 = x, x_2 = x^2, x_3 = x^3, \dots, x_n = x^n$$



## Multiple and Polynomial Regression

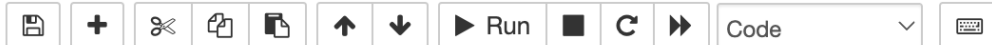
### Learning Goals

- Revisit pandas dataframe
- Explore the data with scatter plot, histogram, probability distribution plot
- Plot the diurnal variation for time series data with the groupby in the boxplot
- Revisit simple linear Regression
- Simple linear Regression with CI
- Multiple regression with Polynomial features

```
In [3]: # import the necessary libraries
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import statsmodels.api as sm
from statsmodels.api import OLS
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score
```



## 1. Extending Linear Regression by Transforming Predictors

Linear regression works great when our features are all continuous and all linearly affect the output. But often real data have more interesting characteristics. Here, we'll look at how we can extend linear regression to handle:

- Categorical predictors, like gender, which have one of a few discrete values
- Interactions between predictors, which let us model how one variable changes the effect of another.

For our dataset, we'll be using the passenger list from the Titanic, which famously sank in 1912. Let's have a look at the data. Some descriptions of the data are at <https://www.kaggle.com/c/titanic/data>, and here's [how seaborn preprocessed it](#).

```
In [2]: # you can also load the dataset from seaborn
titanic = sns.load_dataset("titanic")
titanic.head()
```

```
Out[2]:
```

survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
----------	--------	-----	-----	-------	-------	------	----------	-------	-----	------------	------	-------------	-------	-------



# Dealing with Missing Data

# What is missing data

Often times when data is collected, there are some missing values apparent in the dataset.

This leads to a few questions to consider:

- How does this show up in pandas?
- How does pandas and sklearn handle these NaNs?
- How does this effect our regression modeling?

# How does this show up in pandas?

Try this in your jupyter notebook

```
import pandas as pd
import numpy as np
```

```
df =
pd.DataFrame(np.random.randn(10,6))
# Make a few areas have NaN values
df.iloc[1:3,1] = np.nan
df.iloc[5,3] = np.nan
df.iloc[7:9,5] = np.nan
```

Jupyter Dealing with Missing data-Student Last Checkpoint: 9 minutes ago (e

File Edit View Insert Cell Kernel Widgets Help

+ ✂ 📄 📄 ⬆ ⬆ ▶ Run ■ ↺ ▶▶ Code ▾ 🖨

## Learning Goals

- Identify missing data
- Imputation with KNN
- Imputation with mean
- Imputation with mean linear regression
- Compare the R2 score with teh different imputation method

## Part I. Indentify missing data

```
In [ ]: # generate data set with NaNs
df = pd.DataFrame(np.random.randn(10,6))
# Make a few areas have NaN values
df.iloc[1:3,1] = np.nan
df.iloc[5,3] = np.nan
df.iloc[7:9,5] = np.nan
```

```
In [ ]: df
```

# How does this show up in pandas?

Try this in your jupyter notebook

```
import pandas as pd
import numpy as np

df =
pd.DataFrame(np.random.randn(10,6))
# Make a few areas have NaN values
df.iloc[1:3,1] = np.nan
df.iloc[5,3] = np.nan
df.iloc[7:9,5] = np.nan
```

	df					
	0	1	2	3	4	5
0	0.457697	0.527876	-0.652863	0.274959	0.133250	0.701085
1	0.323433	NaN	0.539954	-0.970977	-0.916636	-1.977056
2	-0.773902	NaN	0.703307	-0.570060	0.223527	-0.076602
3	-0.544289	-0.041683	-0.542055	0.547316	0.392535	1.495068
4	-0.762304	0.445098	-0.802534	-1.235369	-0.507849	0.235933
5	1.120038	2.336353	-0.449025	NaN	0.385721	2.214468
6	3.108624	-0.326424	-1.314199	-1.757547	0.723202	0.763838
7	-1.357699	-0.200107	-0.789109	-1.050255	0.485467	NaN
8	-0.507000	-0.479539	-0.017161	0.454207	1.398188	NaN
9	0.412066	0.356204	-0.059814	-0.323750	0.647593	-1.012747

# How does this show up in pandas?

`df.isnull()`

`isnull()` would return a dataframe like this:

```
In [8]: df.isnull()
```

```
Out[8]:
```

	0	1	2	3	4	5
0	False	False	False	False	False	False
1	False	True	False	False	False	False
2	False	True	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
5	False	False	False	True	False	False
6	False	False	False	False	False	False
7	False	False	False	False	False	True
8	False	False	False	False	False	True
9	False	False	False	False	False	False

# How does this show up in pandas?

`df.isnull().any()`

you can find just the columns that have NaN values

```
In [9]: df.isnull().any()
```

```
Out[9]: 0    False
        1     True
        2    False
        3     True
        4    False
        5     True
        dtype: bool
```

```
In [ ]:
```

One more `.any()` will tell you if any of the above are True

```
In [10]: df.isnull().any().any()
```

```
Out[10]: True
```

# How does this show up in pandas?

To get the number of NaN values in the DataFrame

```
In [13]: df.isnull().sum()
```

```
Out[13]: 0      0  
         1      2  
         2      0  
         3      1  
         4      0  
         5      2  
         dtype: int64
```

```
In [14]: df.isnull().sum().sum()
```

```
Out[14]: 5
```

To check if *any* column has NaNs

```
In [17]: df[1].hasnans
```

```
Out[17]: True
```

# Sources of Missingness

Missing data can arise from various places in data, for example:

- A survey was conducted and values were just randomly missed.
- A respondent chooses not to respond to a specific question.
- You decide to start collecting a new variable partway through the data collection of a study.



# Types of Missingness

There are 3 major types of missingness to be concerned about:

1. **Missing Completely at Random (MCAR)** - the probability of missingness in a variable is the same for all units. Like randomly poking holes in a data set.
2. **Missing at Random (MAR)** - the probability of missingness in a variable depends only on available information (in other predictors).
3. **Missing Not at Random (MNAR)** - the probability of missingness depends on information that has not been recorded and this information also predicts the missing values.

# Missing completely at random (MCAR)

Missing Completely at Random is the best case scenario, and the easiest to handle:

- Examples: a coin is flipped to determine whether an entry is removed. Or when values were just randomly missed when being entered in the computer.
- Effect if you ignore: there is no effect on inferences.
- How to handle: lots of options, but best to impute (more on next slide).

# Missing at random (MAR)

Missing at random is still a case that can be handled.

- Example(s): men and women respond to the question "have you ever felt harassed at work?" at different rates (and may be harassed at different rates).
- Effect if you ignore: inferences are biased
- How to handle: use the information in the other predictors to build a model and 'impute' a value for the missing entry.

**Key: we can fix any biases by modeling and imputing the missing values based on what is observed!**

# Missing Not at Random (MNAR)

Missing Not at Random is the worst case scenario, and impossible to handle:

- Example(s): patients drop out of a study because they experience some really bad side effect that was not measured. Or cheaters are less likely to respond when asked if you've ever cheated.
- Effect if you ignore: there is no effect on inferences
- How to handle: you can 'improve' things by dealing with it like it is MAR, but you [likely] may never completely fix the bias.

# What type of missingness is present?

Can you ever tell based on your data what type of missingness is actually present?

It generally cannot be determined whether data really are missing at random, or whether the missingness depends on unobserved X-variables or the missing data themselves. The problem is that these potential “lurking variables” are unobserved (by definition) and so can never be completely ruled out.

In practice, a dataset with as many X-variables as possible is used so that the ‘missing at random’ assumption is reasonable.

# Handling Missing Data

When encountering missing data, the approach to handling it depends on:

1. whether the missing values are in the response ( $y$ ) or in the predictors ( $X$ ). Generally speaking, it is much easier to handle missingness in predictors.
2. whether the variable is quantitative or categorical.
3. how much missingness is present in the variable. If there is too much missingness, you may be doing more damage than good.

# Naively handling missingness

What is the simplest way to handle missing data?

- Dropping all the points with the missing data

Or we can do Imputation

# Imputation Methods








There are several different approaches to imputing missing values:

1. Plug in the mean or median (quantitative) or most common class (categorical) for all missing values in a variable.
2. Create a new variable that is an indicator of missingness, and include it in any model to predict the response (also plug in zero or the mean in the actual variable).
3. Hot deck imputation: for each missing entry, randomly select an observed entry in the variable and plug it in.
4. Model the imputation: plug in predicted values ( $\hat{y}$ ) from a model based on the other observed predictors.
5. Model the imputation with uncertainty: plug in predicted values plus randomness ( $\hat{y} + \epsilon$ ) from a model based on the other observed predictors.










# Schematic: imputation through modeling

How do we use models to fill in missing data?

X	Y
	1
	?
	0.5
	0.1
	?
	10
	0.03








# Schematic: imputation through modeling

How do we use models to fill in missing data?

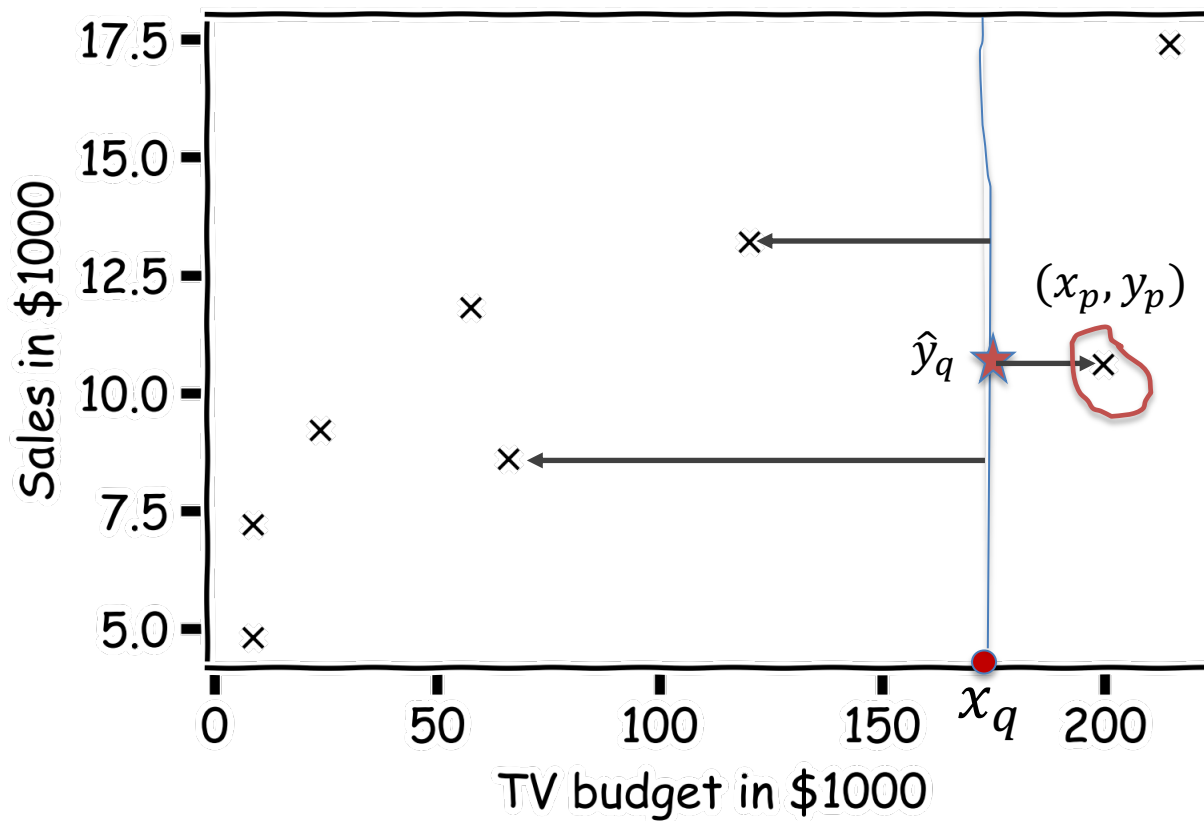
<u>X_train</u>	<u>Y_train</u>	<u>X_test</u>	<u>Y_pred</u>
	1		?
	0.5		
	0.1		
	10		?
	0.03		

# Schematic: imputation through modeling

How do we use models to fill in missing data? Using k-Nearest Neighbors (kNN) for  $k = 2$ ?

X	Y
	1
	? = $(1 + 0.5) / 2$
	0.5
	0.1
	? = $(0.1 + 10) / 2$
	10
	0.03

# k-Nearest Neighbors - kNN



What is  $\hat{y}_q$  at some  $x_q$ ?

Find distances to all other points  $D(x_q, x_i)$

Find the nearest neighbor,  $(x_p, y_p)$

Predict  $\hat{y}_q = y_p$

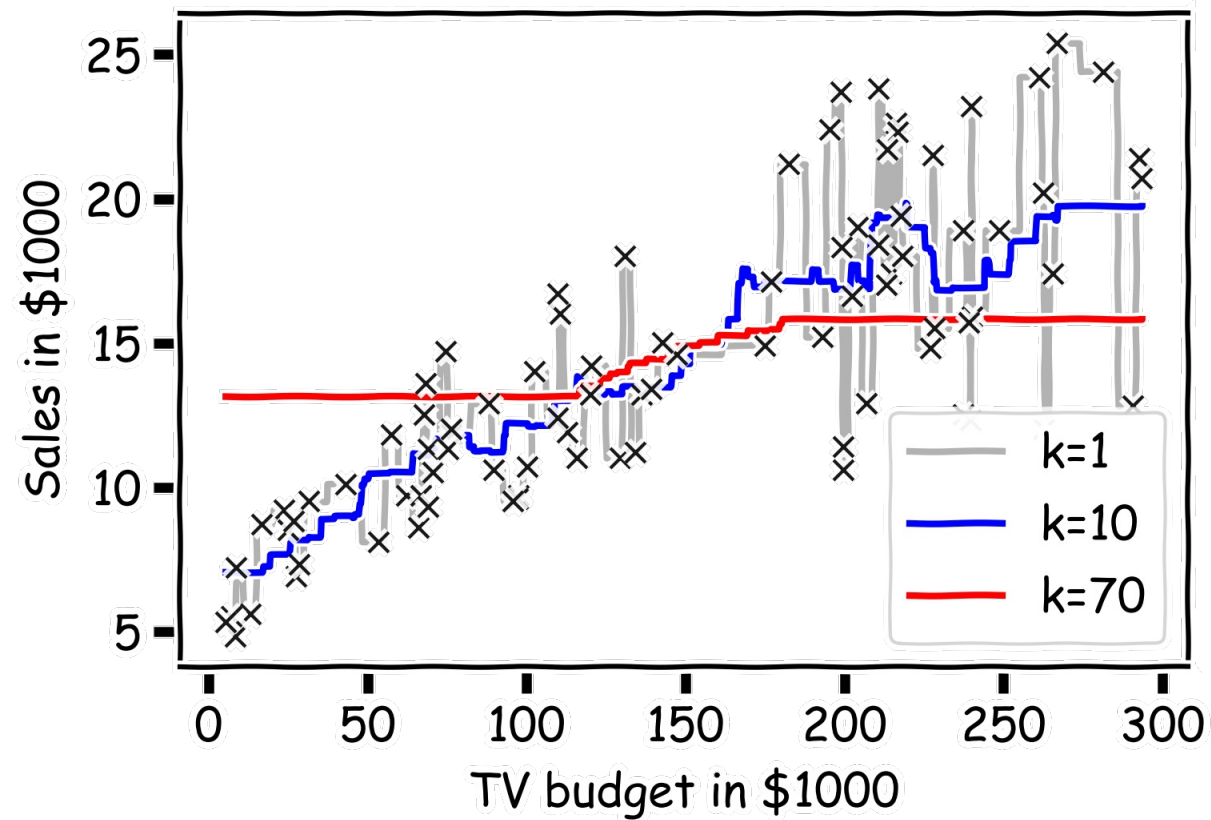
# k-Nearest Neighbors - kNN

For a fixed a value of  $k$ , the predicted response for the  $i$ -th observation is the average of the observed response of the  $k$ -closest observations:

$$\hat{y}_n = \frac{1}{k} \sum_{i=1}^k y_{n_i}$$

where  $\{x_{n1}, \dots, x_{nk}\}$  are the  $k$  observations most similar to  $x_i$  (*similar* refers to a notion of distance between predictors).

# Choice of $k$ matters










# k-Nearest Neighbors – kNN with SK-learn

```
] : # Set kNN parameter:  
k = 100  
  
# Now we can fit the model, predict our variable of interest, and then evaluate our fit:  
neighbors = KNeighborsRegressor(n_neighbors=k)  
  
# Then, we fit the model using x_train as training data and y_train as target values:  
neighbors.fit(train_data[['temp']], train_data['count'])  
  
# Retrieve our predictions:  
prediction_knn = neighbors.predict(test_data[['temp']])
```

# Schematic: imputation through modeling

How do we use models to fill in missing data? Using linear regression?

X	Y
	1
	$? = m \cdot \text{red circle} + b$
	0.5
	0.1
	$? = m \cdot \text{yellow circle} + b$
	10
	0.03

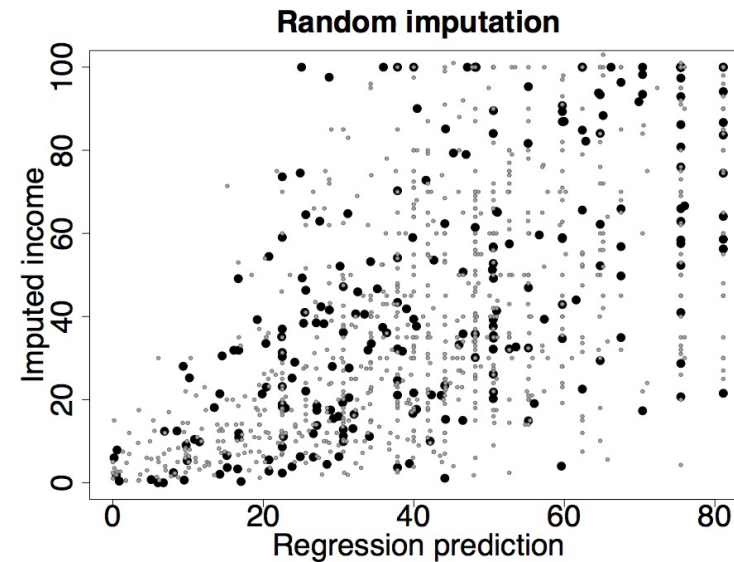
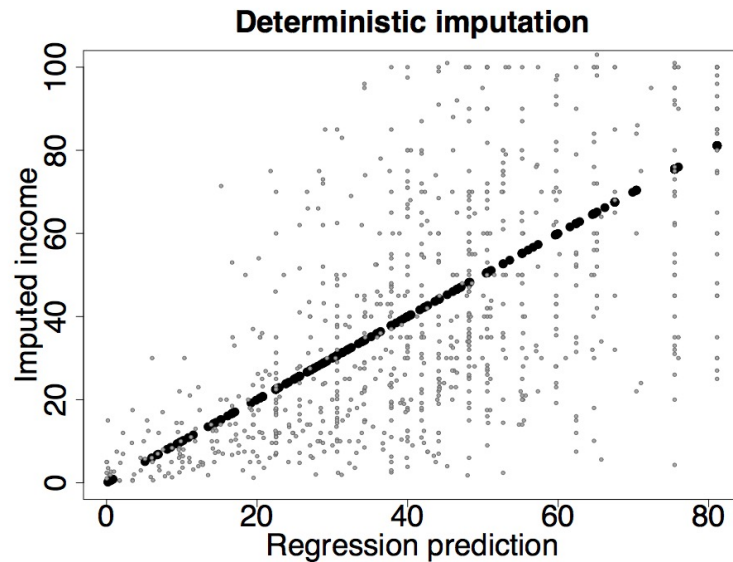
Where  $m$  and  $b$  are computed from the observations (rows) that do not have missingness (we should call them  $b = \beta_0$  and  $m = \beta_1$ ).



# Imputation through modeling with uncertainty

The schematic in the last few slides ignores the fact of imputing with uncertainty. What happens if you ignore this fact and just use the 'best' model to impute values solely on  $\hat{y}$ ?

The distribution of the imputed values will be too narrow and not represent real data (see next slide for illustration). The goal is to impute values that include the uncertainty of the model.



# Imputation through modeling with uncertainty

Recall the model in linear regression:

$$Y = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p + \varepsilon$$

where  $\varepsilon \sim N(0, \sigma^2)$ . How can we take advantage of this model to impute with uncertainty?

# Imputation through modeling with uncertainty

Recall the model in linear regression:

$$Y = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p + \varepsilon$$

where  $\varepsilon \sim N(0, \sigma^2)$ . How can we take advantage of this model to impute with uncertainty?

It's a 3 step process:

1. Fit a model to predict the predictor variable with missingness from all the other predictors.
2. Predict the missing values from the model in the previous part.
3. Add in a measure of uncertainty to this prediction by randomly sampling from a  $N(0, \sigma^2)$  distribution, where  $\sigma^2$  is the mean square error (MSE) from the model.

# Imputation across multiple variables

If only one variable has missing entries, life is easy.

But what if all the predictor variables have a little bit of missingness (with some observations having multiple entries missing)? How can we handle that?

It's an iterative process. Impute  $X_1$  based on  $X_2, \dots, X_p$ . Then impute  $X_2$  based on  $X_1$  and  $X_3, \dots, X_p$ . And continue down the line.



L



## Learning Goals

- Identify missing data
- Imputation with KNN
- Imputation with mean
- Imputation with mean linear regression
- Compare the R2 score with the different imputation method

## Part I. Identify missing data

```
In [ ]: # generate data set with NaNs
df = pd.DataFrame(np.random.randn(10,6))
# Make a few areas have NaN values
df.iloc[1:3,1] = np.nan
df.iloc[5,3] = np.nan
df.iloc[7:9,5] = np.nan
```

```
In [ ]: df
```



School of Energy and Environment

香港城市大學  
City University of Hong Kong

