Machine Learning : 06048203

Feature Engineering

Feature Engineering and Data Preparation

Feature Scaling

Feature Scaling

X1	X2	X3	X4
$ 179.43	56.784	34.6181	3.55
$ 641.87	62.054	47.7306	1.692
$ 556.30	64.13	55.596	1.559
$ 578.47	63.377	52.7121	1.679
$ 591.16	61.553	46.1315	1.984
$ 242.03	58.29	39.2952	2.942
$ 364.66	59.93	42.4628	2.494
$ 190.68	57.271	36.2725	3.419
$ 547.23	63.763	54.1971	1.634
$ 359.69	59.375	41.5105	2.128
$ 438.08	60.484	43.493	2.47
$ 637.17	62.525	49.428	1.725

Feature Scaling

Min-Max scaling Normalization

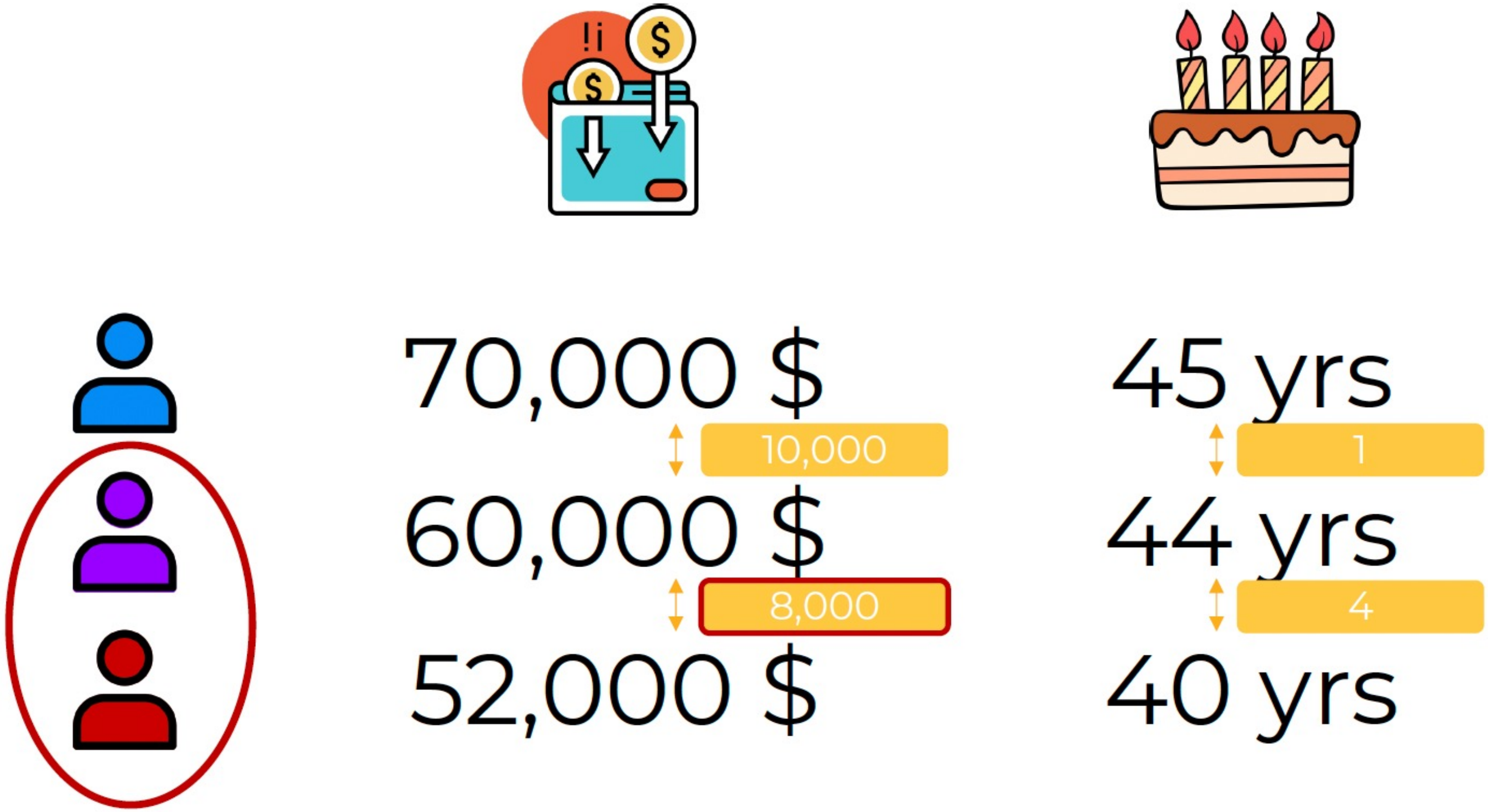
$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

[0 ; 1]

Standard Scaler Normalization

$$X' = \frac{X - \mu}{\sigma}$$

Feature Scaling



Feature Scaling

Min-Max scaling Normalization

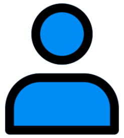
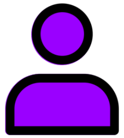
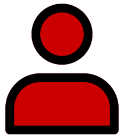
$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

[0 ; 1]

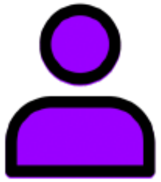
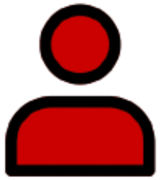
Feature Scaling

		
	70,000 $	45 yrs
	60,000 $	44 yrs
	52,000 $	40 yrs

Feature Scaling

		
	1	45 yrs
	0.444	44 yrs
	0	40 yrs

Feature Scaling

		
	1	1
	0.444	0.75
	0	0

Min-Max scaling Normalization

```python
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
data   = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]
scaler = MinMaxScaler()


xdata = pd.DataFrame(data, columns=['x1', 'x2'])
xdata
```

✓ 0.0s

	x1	x2
0	-1.0	2
1	-0.5	6
2	0.0	10
3	1.0	18

```python
xscale = scaler.fit_transform(xdata)
xscale = pd.DataFrame(xscale, columns=['x1', 'x2'])
xscale
```

✓ 0.0s

	x1	x2
0	0.00	0.00
1	0.25	0.25
2	0.50	0.50
3	1.00	1.00

```python
x = scaler.inverse_transform(xscale)
x = pd.DataFrame(x,  columns=['x1', 'x2'])
x
```

✓ 0.0s

	x1	x2
0	-1.0	2.0
1	-0.5	6.0
2	0.0	10.0
3	1.0	18.0

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

$$X = X'(X_{max} - X_{min}) + X_{min}$$

Standard Scaler Normalization

```python
from sklearn.preprocessing import StandardScaler

import pandas as pd
data   = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]
scaler = StandardScaler()


xdata = pd.DataFrame(data, columns=['x1', 'x2'])
xdata
```

✓ 0.0s

	x1	x2
0	-1.0	2
1	-0.5	6
2	0.0	10
3	1.0	18

```python
xscale = scaler.fit_transform(xdata)
xscale = pd.DataFrame(xscale, columns=['x1', 'x2'])
xscale
```

✓ 0.0s

	x1	x2
0	-1.183216	-1.183216
1	-0.507093	-0.507093
2	0.169031	0.169031
3	1.521278	1.521278

```python
x = scaler.inverse_transform(xscale)
x = pd.DataFrame(x,  columns=['x1', 'x2'])
x
```

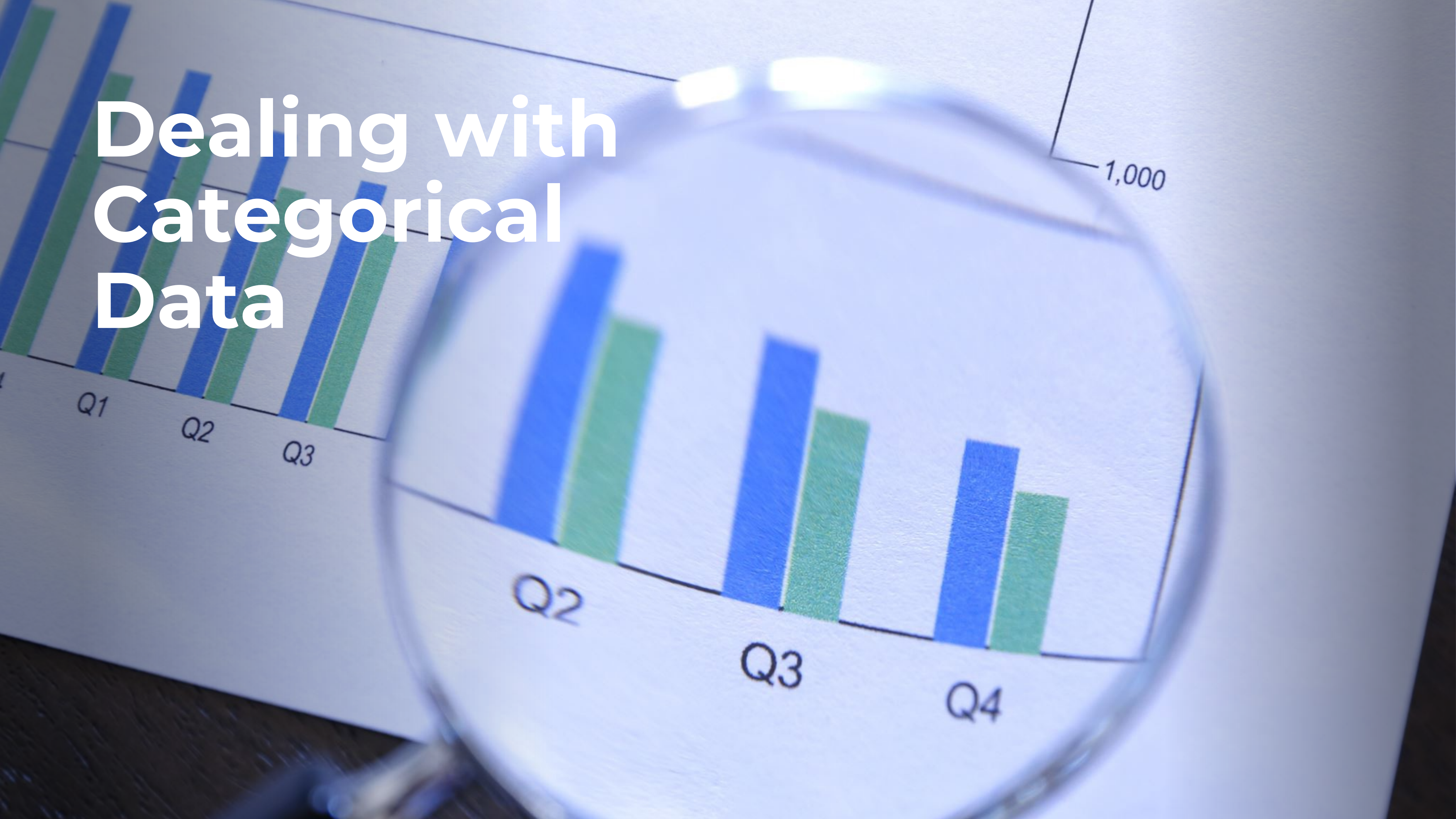
✓ 0.0s

	x1	x2
0	-1.0	2.0
1	-0.5	6.0
2	0.0	10.0
3	1.0	18.0

$$X' = \frac{X - \mu}{\sigma}$$

$$X = \sigma X' + \mu$$

Dealing with Categorical Data



Feature Engineering

- Integer Encoding
  - Directly convert categories into integers 1,2,3…N

Country
USA
MEX
CAN
USA

Feature Engineering

- Integer Encoding
  - Possible issue is implied ordering and relationship (ordinal variable)

Country
USA
MEX
CAN
USA

Country
1
2
3
1

Feature Engineering

- Integer Encoding
  - Pros:
    - Very easy to do and understand.
    - Does not increase number of features.
  - Cons:
    - Implies ordered relationship between categories.

Feature Engineering

- One Hot Encoding (Dummy Variables)
  - Convert categories into individual features that are either 0 or 1

Country
USA
MEX
CAN
USA

Feature Engineering

- One Hot Encoding (Dummy Variables)
  - Convert categories into individual features that are either 0 or 1

Country
USA
MEX
CAN
USA

USA	MEX	CAN
1	0	0
0	1	0
0	0	1
1	0	0

Feature Engineering

- One Hot Encoding (Dummy Variables)
  - No ordered relationship is implied between categories.

Country
USA
MEX
CAN
USA

USA	MEX	CAN
1	0	0
0	1	0
0	0	1
1	0	0

Feature Engineering

- One Hot Encoding (Dummy Variables)
  - We can try to reduce this feature column expansion by creating higher level categories.
  - For example, regions or continents instead of countries.

Feature Engineering

- One Hot Encoding (Dummy Variables)
  - Consider a binary category (only two options):

Vertical Direction
UP
DOWN
UP
DOWN

Feature Engineering

- One Hot Encoding (Dummy Variables)
  - Consider a binary category (only two options):

Vertical Direction
UP
DOWN
UP
DOWN

UP	DOWN
1	0
0	1
1	0
0	1

Feature Engineering

- One Hot Encoding (Dummy Variables)
  - The new columns are duplicate information with inverted encoding.

Vertical Direction
UP
DOWN
UP
DOWN

UP	DOWN
1	0
0	1
1	0
0	1

Feature Engineering

- One Hot Encoding (Dummy Variables)
  - Easily fixed by simply dropping last column.

Vertical Direction
UP
DOWN
UP
DOWN

UP
1
0
1
0

Feature Engineering

- One Hot Encoding (Dummy Variables)
  - This can be extended to more than 2 categories:

Country
USA
MEX
CAN
USA

USA	MEX
1	0
0	1
0	0
1	0

Feature Engineering

- One Hot Encoding (Dummy Variables)
    - Pros:
        - No ordering implied.
    - Cons:
        - Potential to create many more feature columns and coefficients.
        - Dummy variable trap consideration.
        - Not easy to add new categories.

Dealing with Outliers

Outliers

- Often a data set will have a few points that are extreme outliers.
- It’s often better to simply remove these few points from the data set in order to have a more generalized model.

Outliers

- Outlier Considerations
    - Definition of an Outlier
        - Range and Limits
        - Percentage of Data
- These are both very domain dependant!

Outliers

- Outlier Considerations
    - Range and Limits
        - We need to decide what will constitute an outlier with some methodology:
            - InterQuartile Range
            - Standard Deviation
            - Visualized or Domain Limit Value

Outliers

- Outlier Considerations
    - Percentage of Data
        - Keep in mind if a large percentage of your data is being labeled as an outlier, then you actually just have a wide distribution, not outliers!
        - Limit outliers to a few percentage points a most.

Outliers

- Outlier Considerations
    - Utilize visualization plots to be able to see and identify outlier points.
    - Keep in mind, this will create caveats for your future model (e.g. Model not suitable for houses priced over $10 Million)

Outliers

- Keep in mind, there is no 100% correct outlier methodology that will apply to every situation.
- Let’s explore the Ames Data Set for outliers!

Dealing with Missing Data

PART ONE: EVALUATING WHAT IS MISSING

Missing Data

- Make sure you've viewed the "Missing Data" lecture in the pandas section **before** continuing with this series of lectures!
- Many concepts and methods referred to here were explained in those lectures.

Missing Data

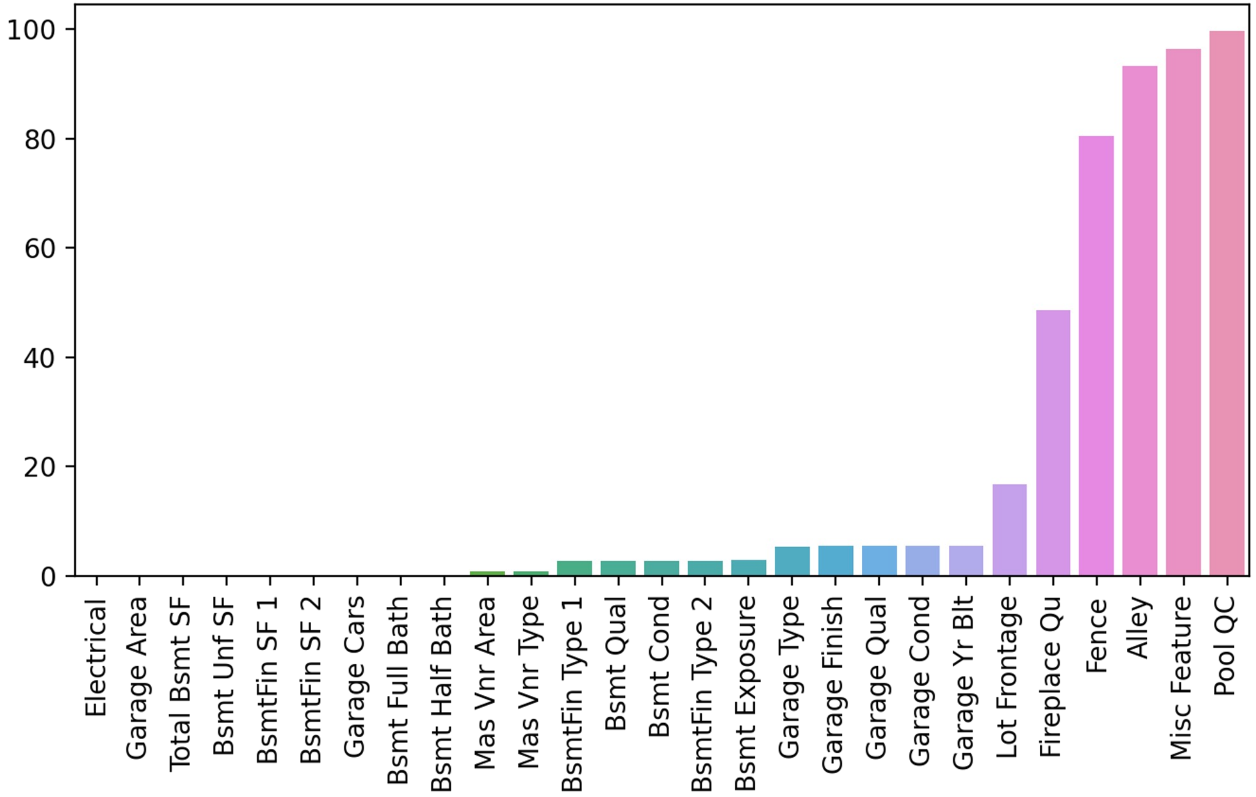
- Working with the Ames data set, in Part One we will focus on evaluating just how much data is missing.

Dealing with Missing Data

PART TWO: FILLING DATA FOR ROWS

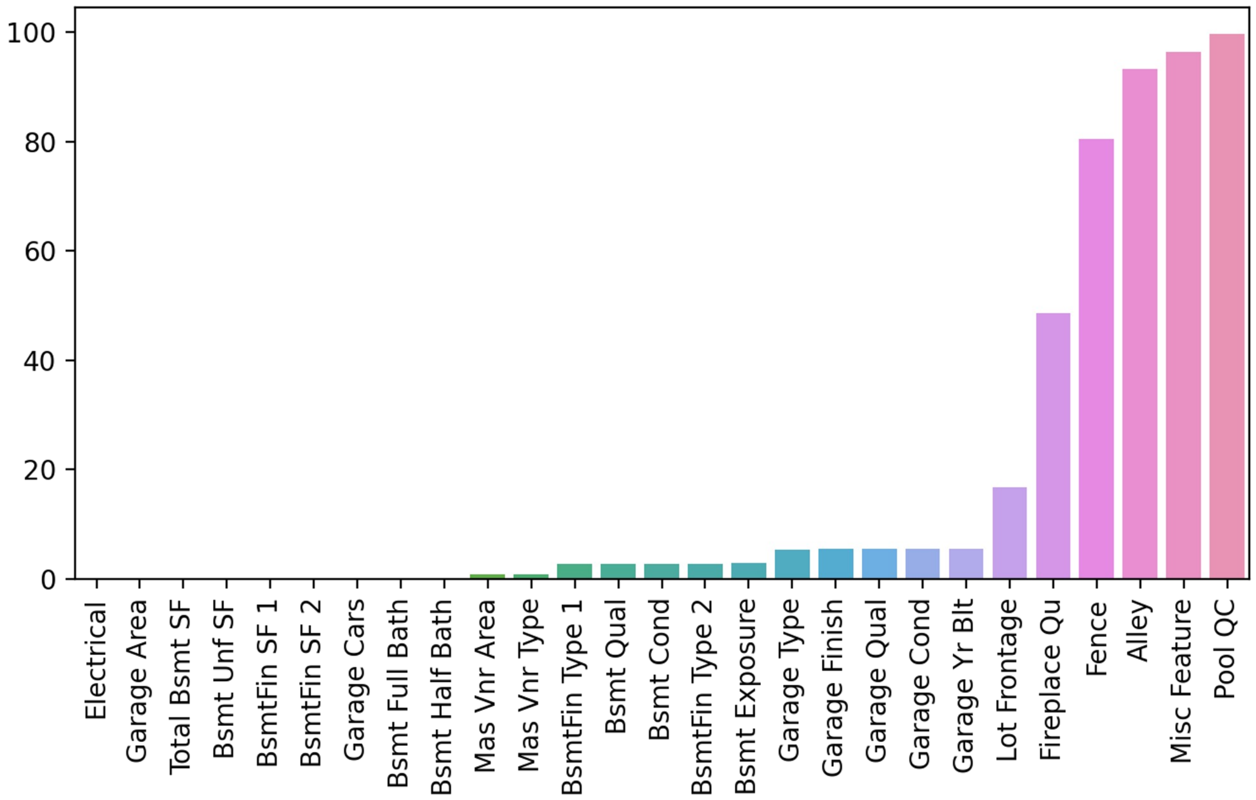
Missing Data

- Recall we just calculated percentage of data missing per feature column:



Missing Data

- Let’s first work on considering features that have a very small percent missing.



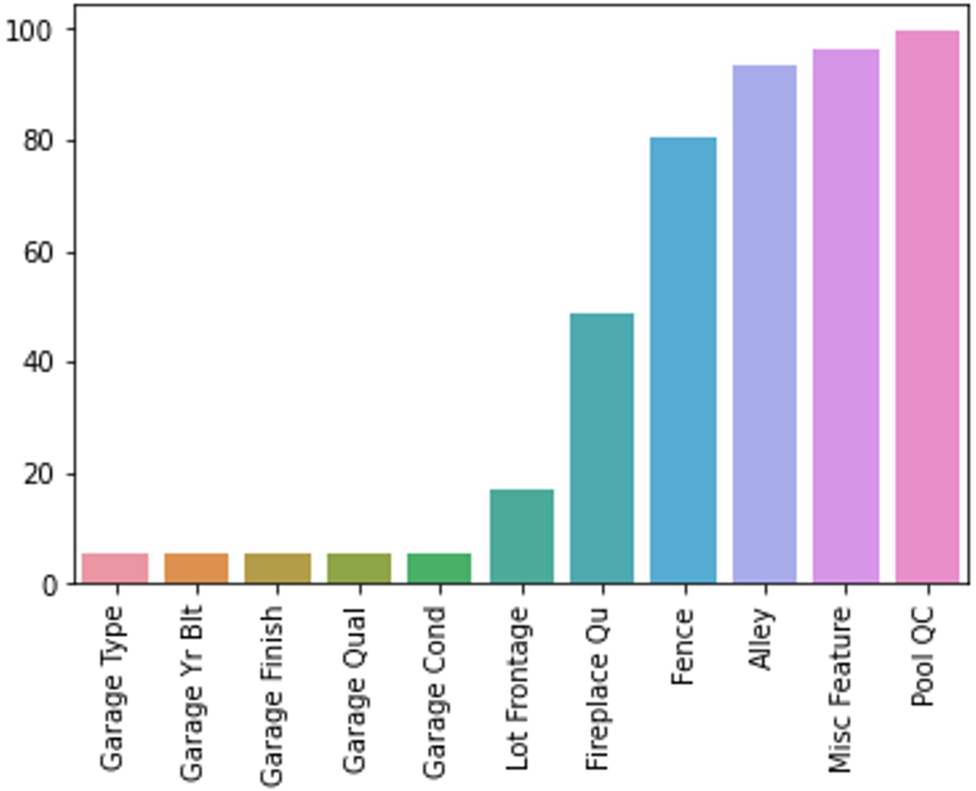
Missing Data

- In the case of just a few rows missing the feature data, we’ll consider either dropping these few rows or filling in with a reasonable assumption based off domain knowledge.
- Let’s jump to the notebook to explore our options!

Dealing with Missing Data

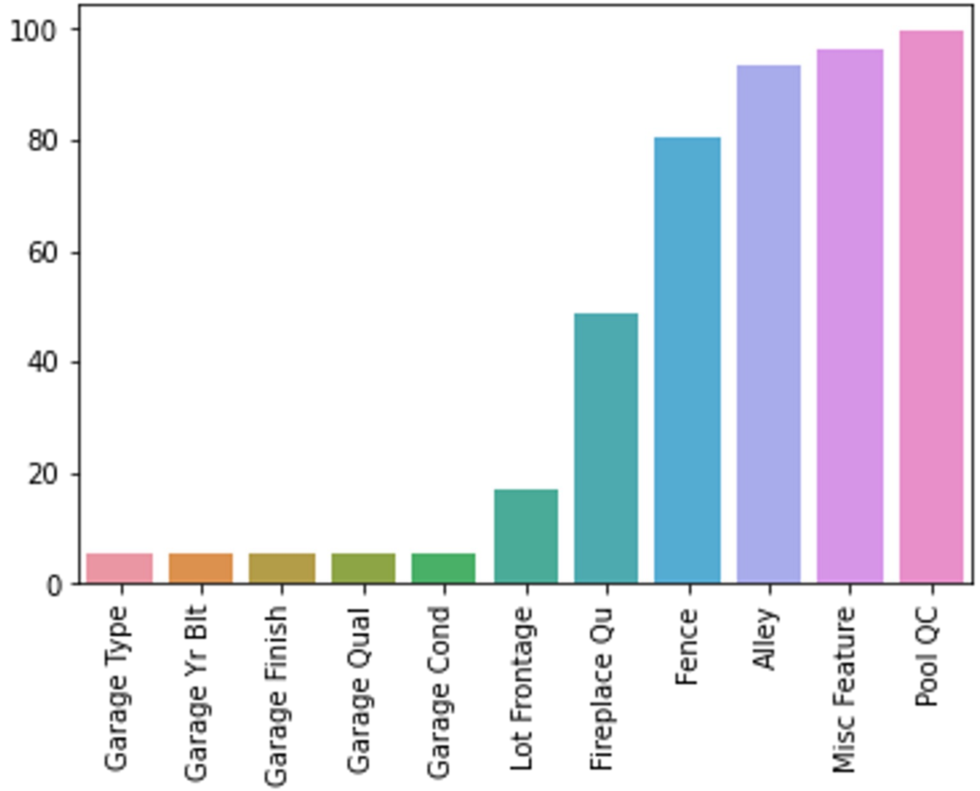
Missing Data

- We are now dealing with missing data that goes beyond our 1% threshold.



Missing Data

- In other words, more than 1% of rows are missing some of these feature values.



Missing Data

- Two main approaches here:
  - Fill in the missing values
  - Drop the feature column
- Let’s consider the pros and cons of each approach…

Missing Data

- Dropping the feature column:
  - Very simple to do.
  - No longer need to worry about that feature in the future.
  - Potential to lose a feature with possible important signal.
  - Should consider drop feature approach when many rows are NaN.

Missing Data

- Filling in the missing feature data:
  - Potentially changing ground truth in data.
  - Must decide on reasonable estimation to filled value.
  - Must apply transformation to all future data for predictions.

Missing Data

- Filling in the missing feature data:
  - Simplest case:
    - Replace all NaN values with a reasonable assumption (e.g. zero if assumed NaN implied zero)
  - Harder cases:
    - Must use statistical methods based on other columns to fill in NaN values.

Missing Data

- Filling in the missing feature data:
  - Statistical Estimation:
    - Dataset about people with some age data missing.
    - Could use current career/education status to fill in data (e.g. people currently in college fill in with 20 yrs)

Missing Data

- Let’s explore both approaches!
    - *Important note!*
        - *Realistically on the Ames data set, many NaN values are probably actually correctly “zero”. But we want to show the methodology for multiple approaches!*

Dealing with Categorical Data

Categorical Data

- We’re going to jump straight to the transformation of the data, but make sure to have watched the section introduction lecture in full for a detailed discussion on dummy variables and one hot encoding!