Data Cleansing and Feature Engineering

Data Cleansing

- Bad data could be missing or wrong
- Remove missing or invalid data
- Remove entire rows or columns if data is missing
- Possibly fix bad values by replacing with average or interpolation
- Scikit-learn allows the dropping of values easily with the `.dropna()` method

```
import pandas as pd

df = pd.DataFrame(
    [
        [1, 1.0, 0, 1],
        [0, 0.5, None, 2],
        [1, 0.2, None, 3],
        [1, 3.3, 0, 4],
        [0, 5.7, 0, "cat"],
        [1, 0.0, None, 6],
        [0, 1.9, 0, 7],
        [1, 2.4, 0, "dog"],
        [None, None, None, 9],
    ]
)

df
# output
     0    1    2    3
0  1.0  1.0  0.0    1
1  0.0  0.5  NaN    2
2  1.0  0.2  NaN    3
3  1.0  3.3  0.0    4
4  0.0  5.7  0.0  cat
5  1.0  0.0  NaN    6
6  0.0  1.9  0.0    7
7  1.0  2.4  0.0  dog
8  NaN  NaN  NaN    9

df.dropna()
# output
     0    1    2    3
0  1.0  1.0  0.0    1
3  1.0  3.3  0.0    4
4  0.0  5.7  0.0  cat
6  0.0  1.9  0.0    7
7  1.0  2.4  0.0  dog
```

Feature Engineering

Feature engineering is useful to create a robust dataset and increase the effectiveness of a model. Generally, they modify or extend the current features in a data set with additional insights or data.

```
import pandas as pd

# Creating a mixed dataset of strings, floats, and date strings
df = pd.DataFrame(
    [
        ["cat", 1.0, "3-2021"],
        ["cat", 0.5, "1-2021"],
        ["dog", 0.2, "5-2021"],
        ["bird", 3.3, "3-2021"],
        ["dog", 5.7, "1-2021"],
        ["dog", 0.0, "2-2021"],
        ["cat", 1.9, "4-2021"],
        ["bird", 2.4, "4-2021"],
        ["bird", 2.4, "5-2021"]
    ],
    columns=["animal", "value", "date"]
)
df.info()


# output
RangeIndex: 9 entries, 0 to 8
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   animal  9 non-null      object
 1   value   9 non-null      float64
 2   date    9 non-null      object
dtypes: float64(1), object(2)
memory usage: 344.0+ bytes
```

Changing Data Types

- `.astype()` function changes column to designated type
- Each type has specific benefits

```
df.loc[:, "animal"] = df["animal"].astype("category")
df.info()


# output
RangeIndex: 9 entries, 0 to 8
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   animal  9 non-null      category
 1   value   9 non-null      float64
 2   date    9 non-null      object
dtypes: category(1), float64(1), object(1)
memory usage: 413.0+ bytes
```

Normalizing Data

- Transforms numerical data to have specific range of values
- Transformations typically have zero mean, meaning their average is 0.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(df[["value"]])
scaler.transform(df[["value"]])


# output
array([[-0.54744332],
       [-0.84071653],
       [-1.01668045],
       [ 0.80161343],
       [ 2.20932483],
       [-1.13398974],
       [-0.01955155],
       [ 0.27372166],
       [ 0.27372166]])
```

Parsing Data Types

- Pandas `to_datetime()` method will parse datetime strings
- Converts strings to datetime objects

```
pd.to_datetime(df.loc[:, "date"])


# output
0   2021-03-01
1   2021-01-01
2   2021-05-01
3   2021-03-01
4   2021-01-01
5   2021-02-01
6   2021-04-01
7   2021-04-01
8   2021-05-01
Name: date, dtype: datetime64[ns]
```

One-hot Encoding

- Required for models that only take numerical data
- Pandas has a one-hot encoding function, `.get_dummies()`
- Converts categorical data to many feature columns

```
# prefix parameter will add the name to column name
pd.get_dummies(df.animal, prefix="animal")


# output
   animal_bird  animal_cat  animal_dog
0            0           1           0
1            0           1           0
2            0           0           1
3            1           0           0
4            0           0           1
5            0           0           1
6            0           1           0
7            1           0           0
8            1           0           0
```

Additional Resources

- Pandas has a good section in their tutorials on **missing data**
- Real Python has a good article on more ways to clean data: **Pythonic Data Cleaning With Pandas and NumPy**