Exercise: Data Cleansing and Feature Engineering

In this exercise, we'll be loading in a dataset that has some problems. In order for us to get it ready for our models, we will apply some of the technics we learned.

Apply these changes to the `data.csv` dataset.

1. Load `data.csv` into a dataframe.
2. Output the table info to see if there are any null values.
3. Remove all null values from the dataframe.
4. Change the `date` column from an object to a `datetime64[ns]` type.
5. Change the `weather` column to a category type.
6. One hot encode the `date` column to year, month, and day.
7. Normalized the columns from the `all_features` list so each feature has a zero mean.
8. Create and save the cleaned dataframe, as well as the train/validation/test dataframes to CSV.

In [1]:

```
import random
from datetime import datetime
import pandas as pd
import numpy as np

from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [2]:

```
# Reading the dataset created by 02_exercise_dataset_creation.ipynb
df = pd.read_csv("data.csv")
```

In [3]:

```
# Always good to check to see if the data looks right
df.head()
```

Out[3]:

	feature0	feature1	feature2	date	weather	target
0	0.274647	-0.603620	0.688897	2021-01-01	sunny	41.269783
1	-0.307691	0.269024	-0.566440	2021-01-01	sunny	-147.974545
2	0.477809	-0.060138	1.974100	2021-01-01	cloudy	204.597486
3	-0.603840	-1.149554	-1.188424	2021-01-01	cloudy	-119.535892
4	0.104714	0.228053	-0.422315	2021-01-01	cloudy	-34.253007

In [4]:

```
# Output general info about the table, notice we have some null values in all of our features
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 6 columns):
feature0    997 non-null float64
feature1    985 non-null float64
feature2    991 non-null float64
date        993 non-null object
weather     989 non-null object
target      1000 non-null float64
dtypes: float64(4), object(2)
memory usage: 47.0+ KB
```

In [5]:

```
# Drop all null values
df = df.dropna()
```

In [6]:

```
# Change the date column to a datetime
df.loc[:, "date"] = pd.to_datetime(df.loc[:, "date"])
# Change weather column to a category
df.loc[:, "weather"] = df["weather"].astype("category")
```

In [7]:

```
# Extract year, month, and day into separate columns
df["year"] = df.date.dt.year
df["month"] = df.date.dt.month
df["day"] = df.date.dt.day
```

In [8]:

```
# One hot encode the weather category to have individual features. Prefix with `weather`
weather_one_hot_df = pd.get_dummies(df.weather, prefix="weather")
```

In [9]:

```
# Add the one hot encoded values back to the df
df[weather_one_hot_df.columns.to_list()] = weather_one_hot_df
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-9-a2426b87e137> in <module>()
      1 # Add the one hot encoded values back to the df
----> 2 df[weather_one_hot_df.columns.to_list()] = weather_one_hot_df

NameError: name 'weather_one_hot_df' is not defined
```

In [10]:

```
# Verify now that are table info has no nulls and correct Dtypes
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 957 entries, 0 to 999
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype         
---  ------          --------------  -----         
 0   feature0        957 non-null    float64       
 1   feature1        957 non-null    float64       
 2   feature2        957 non-null    float64       
 3   date            957 non-null    datetime64[ns]
 4   weather         957 non-null    category      
 5   target          957 non-null    float64       
 6   year            957 non-null    int64         
 7   month           957 non-null    int64         
 8   day             957 non-null    int64         
 9   weather_cloudy  957 non-null    uint8         
 10  weather_rainy   957 non-null    uint8         
 11  weather_sunny   957 non-null    uint8         
dtypes: category(1), datetime64[ns](1), float64(4), int64(3), uint8(3)
memory usage: 71.2 KB
```

In [11]:

```
# These may change if you decided to call your columns different from above
all_features = [
    "feature0",
    "feature1",
    "feature2",
    "year",
    "month",
    "day",
    "weather_cloudy",
    "weather_rainy",
    "weather_sunny",
]
```

In [12]:

```
# Table summary, notice the mean to many of our tables are not zero.
df[all_features].describe()
```

Out[12]:

	feature0	feature1	feature2	year	month	day	weather_cloudy	weather_rainy	weather_sunny
count	957.000000	957.000000	957.000000	957.0	957.000000	957.000000	957.000000	957.000000	957.000000
mean	-0.029455	-0.045588	-0.000638	2021.0	1.993730	15.451411	0.324974	0.163009	0.512017
std	0.998751	0.965487	0.937174	0.0	0.830865	8.717497	0.468610	0.369567	0.500117
min	-3.046143	-3.116857	-2.994613	2021.0	1.000000	1.000000	0.000000	0.000000	0.000000
25%	-0.726712	-0.739936	-0.652761	2021.0	1.000000	8.000000	0.000000	0.000000	0.000000
50%	-0.028529	-0.060138	0.021351	2021.0	2.000000	15.000000	0.000000	0.000000	1.000000
75%	0.610379	0.596906	0.658802	2021.0	3.000000	23.000000	1.000000	0.000000	1.000000
max	3.170975	2.929096	2.680571	2021.0	3.000000	31.000000	1.000000	1.000000	1.000000

In [13]:

```
# Standarize feature values to have a zero mean
scaler = StandardScaler()
scaler.fit(df[all_features])
df.loc[:, all_features] = scaler.transform(df[all_features])
```

In [14]:

```
# Verify our features we are using now all have zero mean
df[all_features].describe()
```

Out[14]:

	feature0	feature1	feature2	year	month	day	weather_cloudy	weather_rainy	weather_sunny
count	9.570000e+02	9.570000e+02	9.570000e+02	957.0	957.000000	9.570000e+02	9.570000e+02	9.570000e+02	9.570000e+02
mean	-1.484938e-17	2.598641e-17	-3.341110e-17	0.0	0.000000	-1.781925e-16	7.053455e-17	1.484938e-17	1.243635e-16
std	1.000523e+00	1.000523e+00	1.000523e+00	0.0	1.000523	1.000523e+00	1.000523e+00	1.000523e+00	1.000523e+00
min	-3.022041e+00	-3.182722e+00	-3.196355e+00	0.0	-1.196644	-1.658614e+00	-6.938474e-01	-4.413123e-01	-1.024329e+00
25%	-6.984945e-01	-7.195453e-01	-6.962042e-01	0.0	-1.196644	-8.552118e-01	-6.938474e-01	-4.413123e-01	-1.024329e+00
50%	9.274150e-04	-1.507826e-02	2.347576e-02	0.0	0.007550	-5.180921e-02	-6.938474e-01	-4.413123e-01	9.762485e-01
75%	6.409693e-01	6.658094e-01	7.040158e-01	0.0	1.211744	8.663652e-01	1.441239e+00	-4.413123e-01	9.762485e-01
max	3.206108e+00	3.082632e+00	2.862448e+00	0.0	1.211744	1.784540e+00	1.441239e+00	2.265969e+00	9.762485e-01

In [15]:

```
# train: 0.8 | test: 0.2
df_train, df_test = train_test_split(df, test_size=0.2, random_state=0)

# train: 0.6 | validation: 0.2
df_train, df_val = train_test_split(df_train, test_size=0.25, random_state=0)

# Final dataset sizes: train: 0.6, validation: 0.2, text: 0.2,
```

In [16]:

```
# Output each shape to confirm the size of train/validation/test
print(f"Train: {df_train.shape}")
print(f"Validation: {df_val.shape}")
print(f"Test: {df_test.shape}")
```

```
Train: (573, 12)
Validation: (192, 12)
Test: (192, 12)
```

In [17]:

```
# Save all clean data, and the train, validation, test data as csv
df.to_csv("data_clean.csv", index=False)
df_train.to_csv("train.csv", index=False)
df_val.to_csv("validation.csv", index=False)
df_test.to_csv("test.csv", index=False)
```

In [ ]:

```

```