Feature Engineering

Numeric Data

https://towardsdatascience.com/understanding-feature-engineering-part-1-continuous-numeric-data-da4e47099a7b

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as spstats
%matplotlib inline
```

Double-click (or enter) to edit

```
poke_df = pd.read_csv('Datasets/Pokemon.csv', encoding='utf-8')
poke_df.head()
```

	#	Name	Type 1	Type 2	Total	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generatio
0	1	Bulbasaur	Grass	Poison	318	45	49	49	65	65	45	
1	2	Ivysaur	Grass	Poison	405	60	62	63	80	80	60	
2	3	Venusaur	Grass	Poison	525	80	82	83	100	100	80	
3	3	VenusaurMega Venusaur	Grass	Poison	625	80	100	123	122	120	80	

Values

Several attributes for the data represent numeric raw values which can be used directly.

```
poke_df[['HP', 'Attack', 'Defense']].head()
```

	HP	Attack	Defense
0	45	49	49
1	60	62	63
2	80	82	83
3	80	100	123
4	39	52	43

Counts

Some situations we need to represent the data as frequencies, counts or occurrences of specific attributes

```
popsong_df = pd.read_csv('Datasets/song_views.csv', encoding='utf-8')
popsong_df.head(10)
```

	user_id	song_id	title	listen_count
0	b6b799f34a204bd928ea014c243ddad6d0be4f8f	SOBONKR12A58A7A7E0	You're The One	2
1	b41ead730ac14f6b6717b9cf8859d5579f3f8d4d	SOBONKR12A58A7A7E0	You're The One	0
2	4c84359a164b161496d05282707cecbd50adbfc4	SOBONKR12A58A7A7E0	You're The One	0
3	779b5908593756abb6ff7586177c966022668b06	SOBONKR12A58A7A7E0	You're The One	0
4	dd88ea94f605a63d9fc37a214127e3f00e85e42d	SOBONKR12A58A7A7E0	You're The	0

Double-click (or enter) to edit

```
watched = np.array(popsong_df['listen_count'])
watched[watched >= 1] = 1
popsong_df['watched'] = watched
```

```
popsong_df.head(10)
```

	user_id	song_id	title	listen_count
0	b6b799f34a204bd928ea014c243ddad6d0be4f8f	SOBONKR12A58A7A7E0	You're The One	2
1	b41ead730ac14f6b6717b9cf8859d5579f3f8d4d	SOBONKR12A58A7A7E0	You're The One	0
2	4c84359a164b161496d05282707cecbd50adbfc4	SOBONKR12A58A7A7E0	You're The One	0
3	779b5908593756abb6ff7586177c966022668b06	SOBONKR12A58A7A7E0	You're The One	0
4	dd88ea94f605a63d9fc37a214127e3f00e85e42d	SOBONKR12A58A7A7E0	You're The	0

Double-click (or enter) to edit

```
items_popularity = pd.read_csv('Datasets/item_popularity.csv',
                               encoding='utf-8')
items_popularity['popularity_scale_10'] = np.array(
                   np.round((items_popularity['pop_percent'] * 10)),
                   dtype='int')
items_popularity['popularity_scale_100'] = np.array(
                  np.round((items_popularity['pop_percent'] * 100)),
                  dtype='int')
items_popularity
```

	item_id	pop_percent	popularity_scale_10	popularity_scale_100
0	it_01345	0.98324	10	98
1	it_03431	0.56123	6	56
2	it_04572	0.12098	1	12
3	it_98021	0.35476	4	35
4	it_01298	0.92101	9	92
5	it_90120	0.81212	8	81
6	it_10123	0.56502	6	57

The features depict the item popularities now both on a scale of 1–10 and on a scale of 1–100.

Double-click (or enter) to edit

```
fcc_survey_df = pd.read_csv('Datasets/2016-FCC-New-Coders-Survey-Data.csv', encoding='utf-8'
fcc_survey_df[['ID.x', 'EmploymentField', 'Age', 'Income']].head()
```

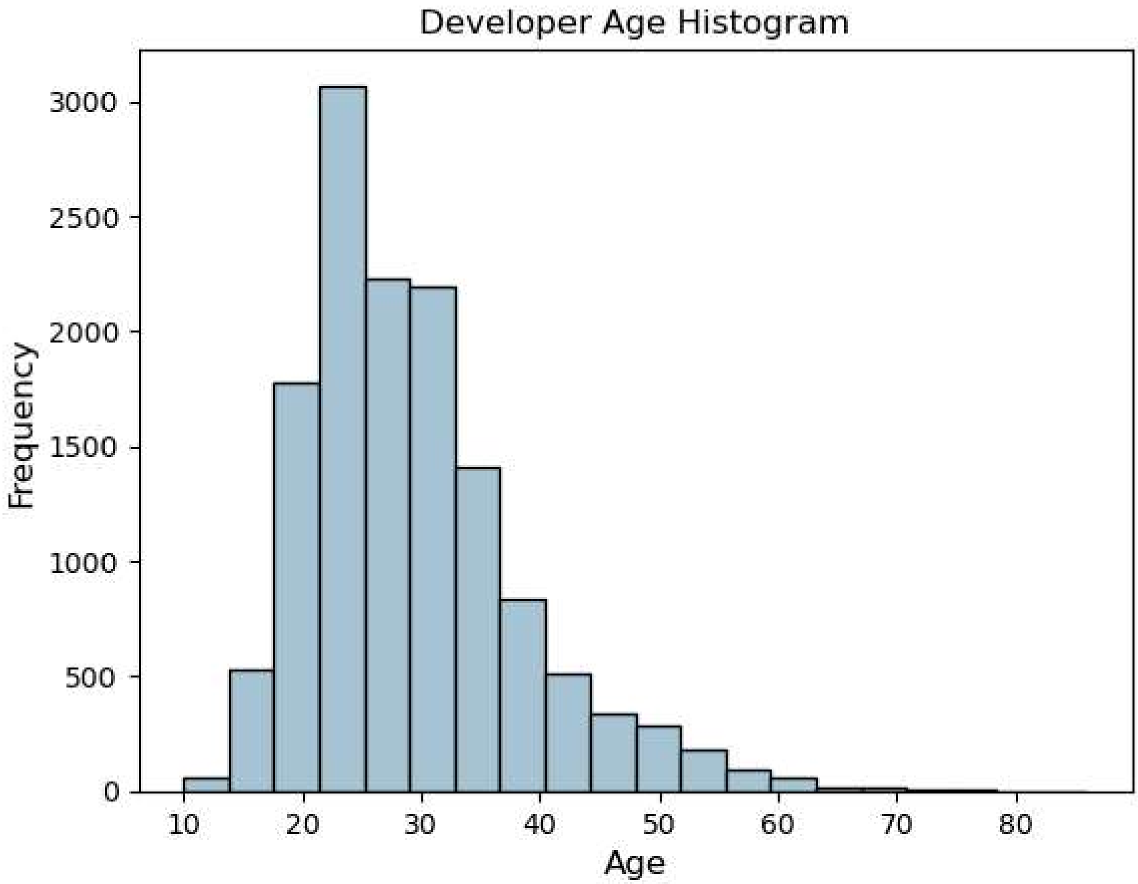
```
/var/folders/jh/t7tw3m2d4pl8tsqlq99htr0w0000gn/T/ipykernel_63061/2611958041.py:1: DtypeW
  fcc_survey_df = pd.read_csv('Datasets/2016-FCC-New-Coders-Survey-Data.csv', encoding='
```

	ID.x	EmploymentField	Age	Income
0	cef35615d61b202f1dc794ef2746df14	office and administrative support	28.0	32000.0
1	323e5a113644d18185c743c241407754	food and beverage	22.0	15000.0
2	b29a1027e5cd062e654a63764157461d	finance	19.0	48000.0
3	04a11e4bcb573a1261eb0d9948d32637	arts, entertainment, sports, or media	26.0	43000.0
4	9368291c93d5d5f5c8cdb1a575e18bec	education	20.0	6000.0

Double-click (or enter) to edit

```
fig, ax = plt.subplots()
num_bins = 20
fcc_survey_df['Age'].hist(bins=num_bins, color='#A9C5D3', edgecolor='black',
                          grid=False)
ax.set_title('Developer Age Histogram', fontsize=12)
ax.set_xlabel('Age', fontsize=12)
ax.set_ylabel('Frequency', fontsize=12)
```

```
Text(0, 0.5, 'Frequency')
```



The corresponding bins for each age have been assigned based on rounding.

```
fcc_survey_df['Age_bin_round'] = np.array(np.floor(
                              np.array(fcc_survey_df['Age']) / 10.))
fcc_survey_df[['ID.x', 'Age', 'Age_bin_round']].iloc[1071:1076]
```

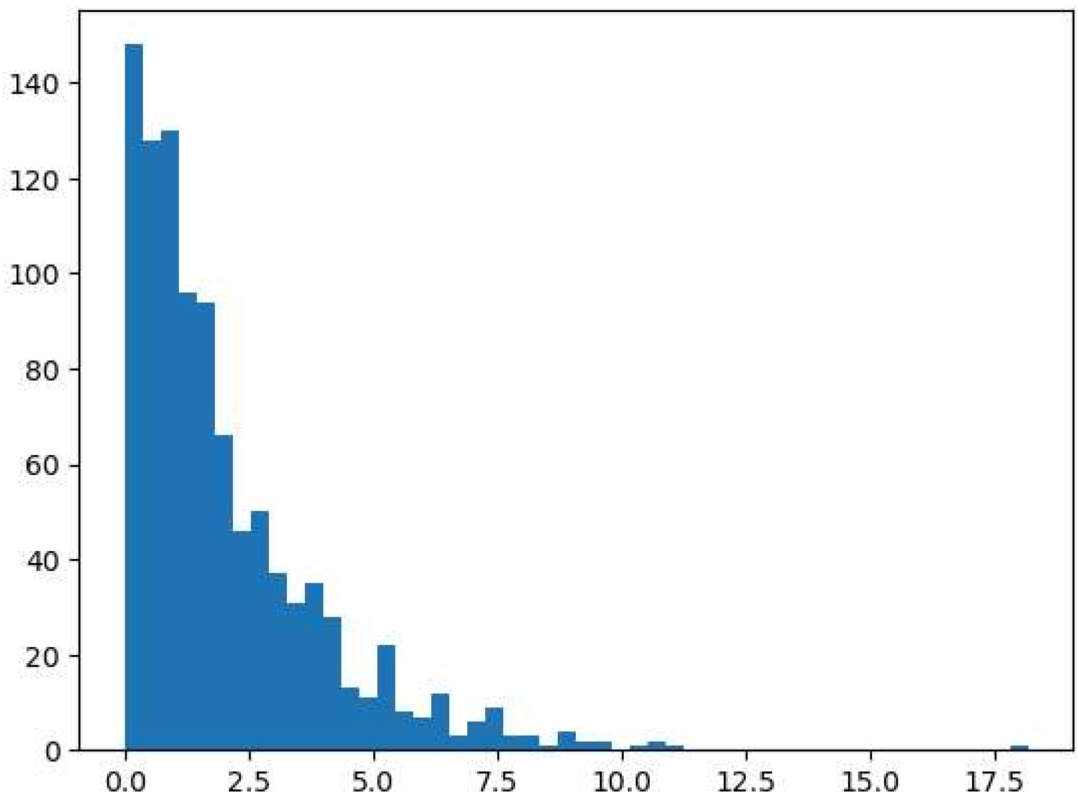
	ID.x	Age	Age_bin_round
1071	16a328da44ceb840863031f35dd3923e	22.0	2.0
1072	54bda0adcf9aa98fa0b3f9b5f608c851	21.0	2.0
1073	0c74ada07ff4ea3f1bf48b4fd79159f9	40.0	4.0
1074	87f04e2ef23e2c517c73e438da1867b1	34.0	3.0
1075	7a2c672065d3802525c733efc1b16219	29.0	2.0

Log Transform

```
import numpy as np
```

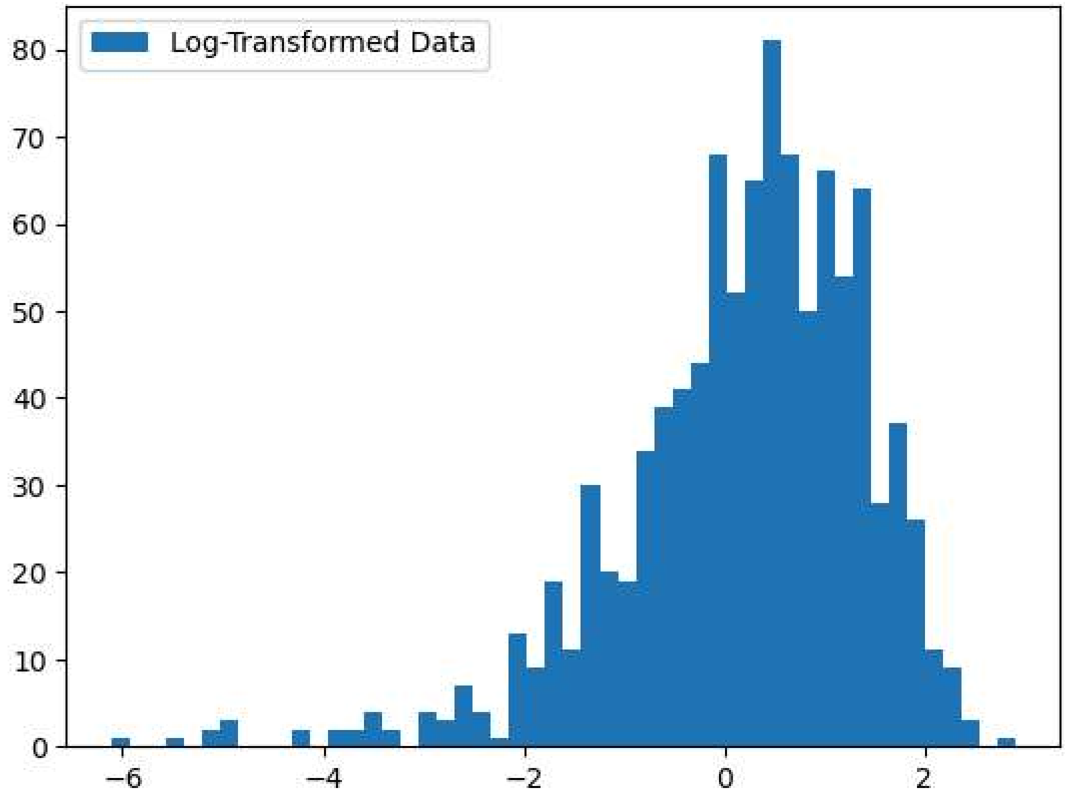
```
# Generate 1000 samples from a right-skewed distribution
data = np.random.exponential(scale=2, size=1000)

# Plot the data to visualize the skew
import matplotlib.pyplot as plt
plt.hist(data, bins=50)
plt.show()
```




```
# Apply log transformation to the data
log_data = np.log(data)

# Plot the original data and the log-transformed data
import matplotlib.pyplot as plt
#plt.hist(data, bins=50, label='Original Data')
plt.hist(log_data, bins=50, label='Log-Transformed Data')
plt.legend()
plt.show()
```



Categorical Data

https://towardsdatascience.com/understanding-feature-engineering-part-2-categorical-data-f54324193e63

Transforming Nominal Attributes

Nominal Data is used to label variables without any order or quantitative value.

- Colour of hair (Blonde, red, Brown, Black, etc.)
- Marital status (Single, Widowed, Married)
- Nationality (Indian, German, American)

```
vg_df = pd.read_csv('Datasets/vgsales.csv', encoding='utf-8')
```

```
vg_df[['Name', 'Platform', 'Year', 'Genre', 'Publisher']].iloc[1:7]
```

	Name	Platform	Year	Genre	Publisher
1	Super Mario Bros.	NES	1985.0	Platform	Nintendo
2	Mario Kart Wii	Wii	2008.0	Racing	Nintendo
3	Wii Sports Resort	Wii	2009.0	Sports	Nintendo
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo
5	Tetris	GB	1989.0	Puzzle	Nintendo
6	New Super Mario Bros.	DS	2006.0	Platform	Nintendo

```
genres = np.unique(vg_df['Genre'])
genres
```

```
array(['Action', 'Adventure', 'Fighting', 'Misc', 'Platform', 'Puzzle',
       'Racing', 'Role-Playing', 'Shooter', 'Simulation', 'Sports',
       'Strategy'], dtype=object)
```

This tells us that we have 12 distinct video game genres. We can now generate a label encoding scheme for mapping each category to a numeric value by leveraging scikit-learn.

```
from sklearn.preprocessing import LabelEncoder
gle = LabelEncoder()
genre_labels = gle.fit_transform(vg_df['Genre'])
genre_mappings = {index: label for index, label in enumerate(gle.classes_)}
genre_mappings

genre_labels
```

```
array([10,  4,  6, ...,  6,  5,  4])
```

```
vg_df['GenreLabel'] = genre_labels
vg_df[['Name', 'Platform', 'Year', 'Genre', 'GenreLabel']].head(10)
```

	Name	Platform	Year	Genre	GenreLabel
0	Wii Sports	Wii	2006.0	Sports	10
1	Super Mario Bros.	NES	1985.0	Platform	4
2	Mario Kart Wii	Wii	2008.0	Racing	6
3	Wii Sports Resort	Wii	2009.0	Sports	10
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	7
5	Tetris	GB	1989.0	Puzzle	5
6	New Super Mario Bros.	DS	2006.0	Platform	4
7	Wii Play	Wii	2006.0	Misc	3
8	New Super Mario Bros. Wii	Wii	2009.0	Platform	4
9	Duck Hunt	NES	1984.0	Shooter	8

Transforming Ordinal Attributes

Ordinal attributes are categorical attributes with a sense of order amongst the values.

- Economic Status (High, Medium, and Low)
- Education Level (Higher, Secondary, Primary)

```
poke_df = pd.read_csv('Datasets/Pokemon.csv', encoding='utf-8')
# poke_df = poke_df.sample(random_state=1,frac=1).reset_index(drop=True)
print(poke_df.head())
np.unique(poke_df['Generation'])
```

```
     #                    Name  Type 1  Type 2  Total  HP  Attack  Defense  \
0    6  CharizardMega Charizard Y    Fire  Flying    634  78     104       78
1  460               Abomasnow   Grass     Ice    494  90      92       75
2  161                 Sentret  Normal     NaN    215  35      46       34
3  667                  Litleo    Fire  Normal    369  62      50       58
4  224               Octillery   Water     NaN    480  75     105       75

   Sp. Atk  Sp. Def  Speed  Generation  Legendary
0      159      115    100           1      False
1       92       85     60           4      False
2       35       45     20           2      False
3       73       54     72           6      False
4      105       75     45           2      False
array([1, 2, 3, 4, 5, 6])
```

In general, there is no generic module or function to map and transform these features into numeric representations based on order automatically. Hence we can use a custom encoding\mapping scheme.

```
gen_ord_map = {'Gen 1': 1, 'Gen 2': 2, 'Gen 3': 3,'Gen 4': 4, 'Gen 5': 5, 'Gen 6': 6}

poke_df['GenerationLabel'] = poke_df['Generation'].map(gen_ord_map)
poke_df[['Name', 'Generation', 'GenerationLabel']].iloc[4:10]
```

	Name	Generation	GenerationLabel
4	Octillery	2	NaN
5	Helioptile	6	NaN
6	Dialga	4	NaN
7	DeoxysDefense Forme	3	NaN
8	Rapidash	1	NaN
9	Swanna	5	NaN

```
poke_df[['Name', 'Generation', 'Legendary']].iloc[4:10]
```

	Name	Generation	Legendary
4	Octillery	2	False
5	Helioptile	6	False
6	Dialga	4	True
7	DeoxysDefense Forme	3	True
8	Rapidash	1	False
9	Swanna	5	False

One-Hot Encoding

```
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
gen_le = LabelEncoder()
leg_le = LabelEncoder()
leg_labels = leg_le.fit_transform(poke_df['Legendary'])
poke_df['Lgnd_Label'] = leg_labels
poke_df_sub = poke_df[['Name', 'Generation', 'Legendary', 'Lgnd_Label', 'Type 1']]
poke_df_sub.iloc[4:10]
```

	Name	Generation	Legendary	Lgnd_Label	Type 1
4	Octillery	2	False	0	Water
5	Helioptile	6	False	0	Electric
6	Dialga	4	True	1	Steel
7	DeoxysDefense Forme	3	True	1	Psychic
8	Rapidash	1	False	0	Fire
9	Swanna	5	False	0	Water

```
# encode legendary status labels using one-hot encoding scheme
leg_ohe = OneHotEncoder()
leg_feature_arr = leg_ohe.fit_transform(poke_df[['Lgnd_Label']]).toarray()
leg_feature_labels = ['Legendary_'+str(cls_label)  for cls_label in leg_le.classes_]
leg_features = pd.DataFrame(leg_feature_arr, columns=leg_feature_labels)
poke_df_sub.head(10)
```

	Name	Generation	Legendary	Lgnd_Label	Type 1
0	CharizardMega Charizard Y	1	False	0	Fire
1	Abomasnow	4	False	0	Grass
2	Sentret	2	False	0	Normal
3	Litleo	6	False	0	Fire
4	Octillery	2	False	0	Water
5	Helioptile	6	False	0	Electric
6	Dialga	4	True	1	Steel
7	DeoxysDefense Forme	3	True	1	Psychic
8	Rapidash	1	False	0	Fire
9	Swanna	5	False	0	Water

```
gen_onehot_features = pd.get_dummies(poke_df['Lgnd_Label'])
pd.concat([poke_df[['Name', 'Lgnd_Label']], gen_onehot_features], axis=1).head(10)
```

	Name	Lgnd_Label	0	1
0	CharizardMega Charizard Y	0	1	0
1	Abomasnow	0	1	0
2	Sentret	0	1	0
3	Litleo	0	1	0
4	Octillery	0	1	0
5	Helioptile	0	1	0
6	Dialga	1	0	1
7	DeoxysDefense Forme	1	0	1
8	Rapidash	0	1	0
9	Swanna	0	1	0

```
print(poke_df['Type 1'])
```

```
0          Fire
1         Grass
2        Normal
3          Fire
4         Water
         ...
795      Normal
796        Rock
797    Fighting
798      Normal
799      Poison
Name: Type 1, Length: 800, dtype: object
```

```
gen_onehot_features = pd.get_dummies(poke_df['Type 1'])
pd.concat([poke_df[['Name', 'Type 1']], gen_onehot_features], axis=1).head(10)
```

	Name	Type 1	Bug	Dark	Dragon	Electric	Fairy	Fighting	Fire	Flying	Gho
0	CharizardMega Charizard Y	Fire	0	0	0	0	0	0	1	0	
1	Abomasnow	Grass	0	0	0	0	0	0	0	0	

Dummy Coding Scheme

3	Litleo	Fire	0	0	0	0	0	0	1	0

```
gen_dummy_features = pd.get_dummies(poke_df['Type 1'], drop_first=True)
pd.concat([poke_df[['Name', 'Type 1']], gen_dummy_features], axis=1).head(10)
```

	Name	Type 1	Dark	Dragon	Electric	Fairy	Fighting	Fire	Flying	Ghost
0	CharizardMega Charizard Y	Fire	0	0	0	0	0	1	0	0
1	Abomasnow	Grass	0	0	0	0	0	0	0	0
2	Sentret	Normal	0	0	0	0	0	0	0	0
3	Litleo	Fire	0	0	0	0	0	1	0	0
4	Octillery	Water	0	0	0	0	0	0	0	0
5	Helioptile	Electric	0	0	1	0	0	0	0	0
6	Dialga	Steel	0	0	0	0	0	0	0	0
7	DeoxysDefense Forme	Psychic	0	0	0	0	0	0	0	0
8	Rapidash	Fire	0	0	0	0	0	1	0	0

+ Code + Text

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.