hn325
Hong Noh

Assignment 2

Question 2

2.1 Please refer to assign2.py

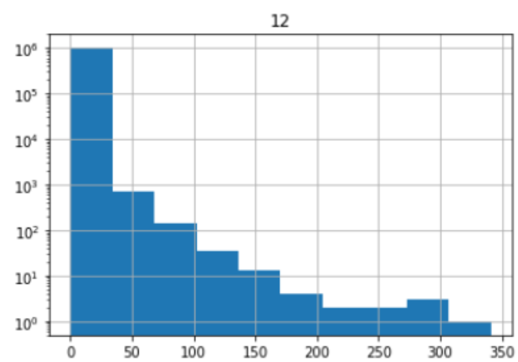
2.2 Feature Summary Statistics

Integer Feature Histograms:

Integer 12

```
# Investigate Integer data 12:
output_dict12 = summ_stats(train1M, 12)
print(output_dict12)
```

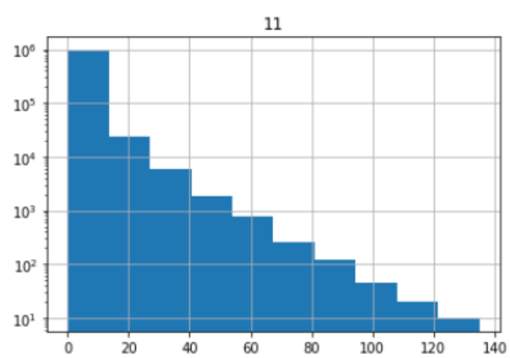
```
{'desc': count    1000000.000000
mean           0.228617
std            2.402548
min            0.000000
25%            0.000000
50%            0.000000
75%            0.000000
max          341.000000
Name: 12, dtype: float64, 'skew': 34.507226529423633, 'kurt': 2281.0104004716791}
```



Integer 11

```
# Investigate Integer data 11:
output_dict11 = summ_stats(train1M, 11)
print(output_dict11)
```

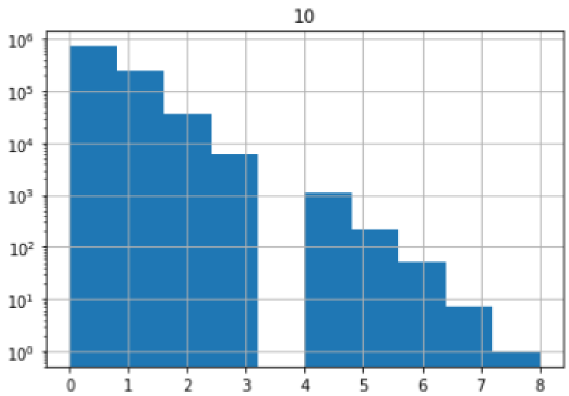
```
{'desc': count    1000000.000000
mean           2.616465
std            5.136210
min            0.000000
25%            0.000000
50%            1.000000
75%            3.000000
max          135.000000
Name: 11, dtype: float64, 'skew': 6.1598226499925186, 'kurt': 61.013681890031656}
```



Integer 10

```
# Investigate Integer data 10:
output_dict10 = summ_stats(train1M, 10)
print(output_dict10)
```

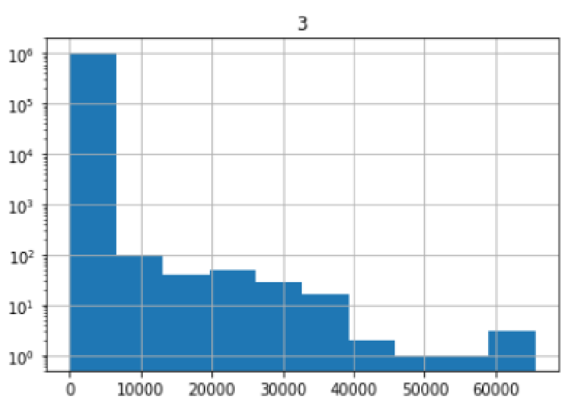
```
{'desc': count    1000000.000000
mean           0.337898
std            0.591993
min            0.000000
25%            0.000000
50%            0.000000
75%            1.000000
max            8.000000
Name: 10, dtype: float64, 'skew': 1.9645423846699259, 'kurt': 5.1374736539093515}
```



Integer 3

```
# Investigate Integer data 3:
output_dict3 = summ_stats(train1M, 3)
print(output_dict3)
```

```
{'desc': count    1000000.000000
mean          21.236923
std          346.876275
min            0.000000
25%            1.000000
50%            4.000000
75%           13.000000
max        65535.000000
Name: 3, dtype: float64, 'skew': 87.041821117926361, 'kurt': 9694.255938490307}
```



For the full syntax of the summ_stats method please refer to the appendix.

hn325
Hong Noh

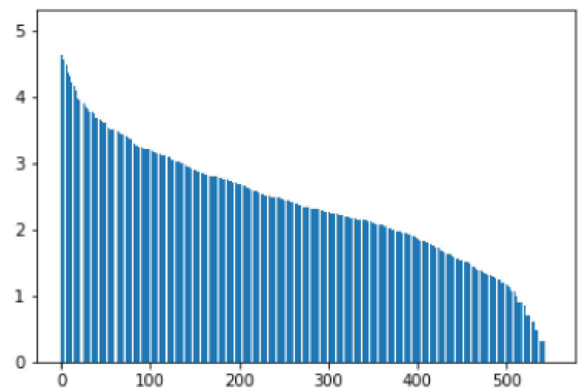
Categorical Feature Histograms:

Categorical Feature 15

```python
# Investigate Categorical data 15:
data15 = investigate_cat(15, train1M)
n = 9000
plot_data15 = {}
for i in range(0,len(data15['Counts'])):
    if (i<9000):
        plot_data15.update({i:data15['Counts'][i]})

plot_dict(plot_data15)
print(data15['Padding Percentage'])
print(data15['Unique Values'])
```

```
0.0
551
```

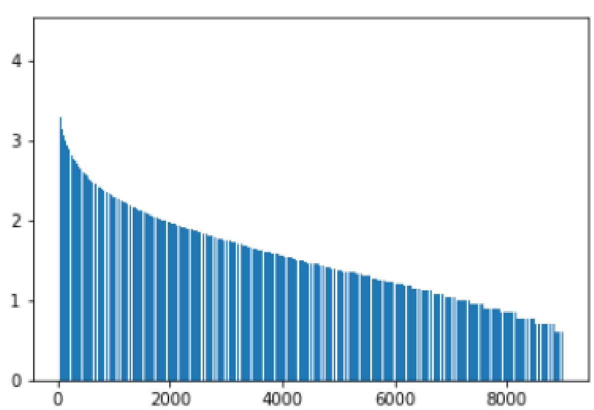


Categorical Feature 20

```python
# Investigate Categorical data 20:
data20 = investigate_cat(20, train1M)
n = 9000
plot_data20 = {}
for i in range(0,len(data20['Counts'])):
    if (i<9000):
        plot_data20.update({i:data20['Counts'][i]})

plot_dict(plot_data20)
print(data20['Padding Percentage'])
print(data20['Unique Values'])
```

```
0.0
11217
```



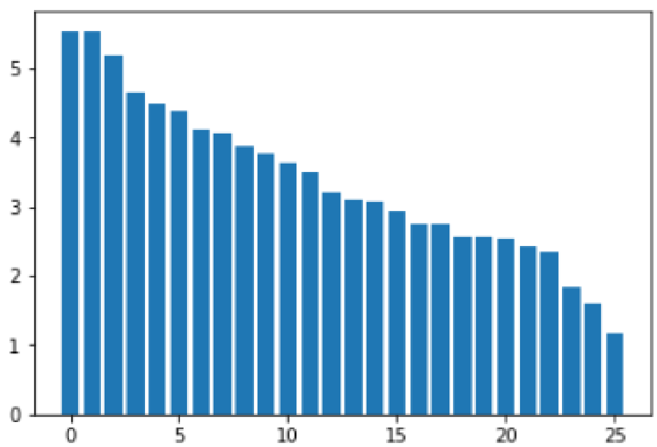
hn325
Hong Noh

Categorical Feature 27

```
# Investigate Categorical data 27:
data27 = investigate_cat(27, train1M)
n = 9000
plot_data27 = {}
for i in range(0,len(data27['Counts'])):
    if (i<9000):
        plot_data27.update({i:data27['Counts'][i]})

plot_dict(plot_data27)
print(data27['Padding Percentage'])
print(data27['Unique Values'])
```

```
0.0
26
```

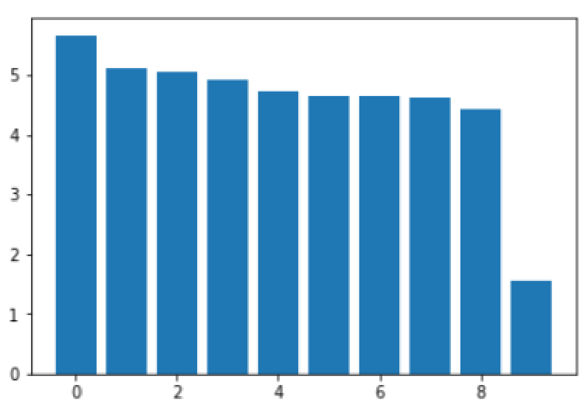


Categorical Feature 30

```
# Investigate Categorical data 30:
data30 = investigate_cat(30, train1M)
n = 9000
plot_data30 = {}
for i in range(0,len(data30['Counts'])):
    if (i<9000):
        plot_data30.update({i:data30['Counts'][i]})

plot_dict(plot_data30)
print(data30['Padding Percentage'])
print(data30['Unique Values'])
```

```
0.0
10
```



Where the full syntax of investigate_cat and plot_dict are available in the appendix.

2.3 Feature Selection and Encoding

When considering categorical features, I observed the following statistics:

- Number of missing values (padded as zeros in the preprocessing)
- Number of unique values the categorical feature can take

Features were dropped if *any* values were missing *or* if the number of unique values the feature can take exceeded 800. Since the intention was to use one hot encoding with 20 bits, we want to minimize information lost in the encoding process. With features with a high number of unique values, only the top 20 of feature values will be encoded, with the remainder grouped as a separate category. I believe this results in such high loss of information that it is not worth including in the encoding process.

Categorical Feature Column Number (zero start)	Missing Values %	Unique Feature Values	Select/Reject	Select Reject Reason
14	0	1249	Reject	High unique values
15	0	551	Select	Low missing values, low unique values
16	3.3853	362872	Reject	High unique values
17	3.3853	141168	Reject	High unique values
18	0	274	Select	Low missing values, low unique values
19	12.159	16	Reject	High missing values
20	0	11217	Reject	High unique values
21	0	557	Select	Low missing values, low unique values
22	0	3	Select	Low missing values, low unique values
23	0	31849	Reject	High unique values
24	0	4916	Reject	High unique values
25	3.3853	322871	Reject	High unique values
26	0	3154	Reject	High unique values
27	0	26	Select	Low missing values, low unique values
28	0	9516	Reject	High unique values
29	3.3853	246473	Reject	High unique values
30	0	10	Select	Low missing values, low unique values
31	0	4093	Reject	High unique values
32	43.9476	1855	Reject	High missing values, high unique values
33	43.9476	4	Reject	High missing values
34	3.3853	291000	Reject	High unique values
35	76.2392	16	Reject	High missing values
36	0	15	Select	Low missing values, low unique values
37	3.3853	44970	Reject	High unique values
38	43.9476	73	Reject	High missing values
39	43.9476	32484	Reject	High missing values, high unique values

Upon observation it is also interesting to see that some features have equal numbers of missing values (one group highlighted in yellow, the other in orange). These features also beckon a rejection decision based on the possibility of high correlation.

This results in the resulting categorical feature set of columns: [14, 17, 20, 21, 26, 29, 35] for the training data (with zero start counting of columns). This is for training data *excluding* the training target.

For the categorical features selected above, I take the 20 most occurring feature values and any other values I group into a feature value of ‘Others’. Once I reduce the number of values the feature can take to these 21 values (the 20 top most occurring plus ‘Others’), I conduct label encoding into integers using LabelEncoder and subsequently apply a fit transform to encode each feature value into 21-bit lists. This information will be stored in the output of the preprocess_cat_data function. To find the *encoded* values from the fit transform, we can use the.toarray() function.

Appendix: Calculating Summary Statistics and Plotting Histograms of Integer Features

```
def summ_stats(input_data, col_ind):
    output_dict = {}
    desc = input_data[col_ind].describe()
    skew = input_data[col_ind].skew(axis=0)
    kurt = input_data[col_ind].kurtosis(axis=0)
    output_dict['desc'] = desc
    output_dict['skew'] = skew
    output_dict['kurt'] = kurt
    %matplotlib inline
    train1M.hist(column=col_ind, log=True)
    return output_dict
```

Appendix: Investigating Categorical Features

```
def investigate_cat(ind, data):
    import matplotlib.pyplot as plt
    cat_kc = data[ind].value_counts()

    cat_ind = cat_kc.index.tolist()
    cat_count = np.log10(cat_kc.values.tolist())

    count_dict = {}

    for i in range(0,len(cat_ind)):# if len(cat_ind)<9000 else 9000)):
        count_dict.update({i:cat_count[i]})

    output_dict = {}

    if 0 in cat_ind:
        missing_vals = cat_kc.values[cat_ind.index(0)]
    else:
        missing_vals = 0

    pct_padding = missing_vals/len(data)

    output_dict['Counts'] = count_dict
    output_dict['Value Counts'] = cat_kc
    output_dict['Padding Percentage'] = pct_padding
    output_dict['Unique Values'] = len(cat_ind)

    return output_dict
```

Appendix: Plotting Categorical Features

```
def plot_dict(input_dict):
    %matplotlib inline
    import matplotlib.pyplot as plt
    lists = sorted(input_dict.items())

    x,y = zip(*lists)
    plt.bar(x,y)
```