

## 2.2 Problem 2

Load the auto-mpg sample dataset into Python using a Pandas dataframe. The horsepower feature has a few missing values with a? - replace these with a NaN from NumPy, and calculate summary statistics for each numerical column.

How do the summary statistics vary when excluding the NaNs, vs. imputing them with the mean (Hint: Use an Imputer from Scikit) - can we do better than just using the overall sample mean?

### **Answer:**

Summary statistics excluding NaN –

```
count      398
unique       94
top        150
freq         22
Name: horsepower, dtype: object
```

Summary statistics after imputing -

```
count      392.000000
mean       104.469388
std        38.491160
min         46.000000
25%                NaN
50%                NaN
75%                NaN
max        230.000000
Name: horsepower, dtype: float64
```

Here, the count has changed as the missing values has been replaced by the mean. Yes, we can replace the values by using median or mode and improve the overall result but standard deviation increases.

```
In [14]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
In [15]: from sklearn.preprocessing import Imputer
```

```
In [16]: auto = pd.read_table(r'C:\Python34\Lib\site-packages\Orange\datasets\auto-mpg.tab', sep = '\t').ix[2:]
```

```
In [17]: auto2 = auto['horsepower']
```

```
In [18]: auto2 = pd.to_numeric(auto2, errors='coerce')
```

```
In [19]: auto2 = auto2.reshape(-1,1)
```

```
In [20]: imp = Imputer(missing_values='NaN', strategy='mean', axis=0)
```

```
In [21]: imp.fit(auto2)
```

```
Out[21]: Imputer(axis=0, copy=True, missing_values='NaN', strategy='mean', verbose=0)
```

```
In [22]: auto_mean = imp.fit_transform(auto2)
```

```
In [23]: auto['horsepower'] = auto_mean
```

```
In [24]: auto.describe()
```

```
Out[24]:
```

	horsepower
count	398.000000
mean	104.469388
std	38.199187
min	46.000000
25%	76.000000
50%	95.000000
75%	125.000000
max	230.000000

```
In [25]: imp = Imputer(missing_values='NaN', strategy='median', axis=0)
imp.fit(auto2)
auto_median = imp.fit_transform(auto2)
auto['horsepower'] = auto_median
auto.describe()
```

```
Out[25]:
```

	horsepower
count	398.000000
mean	104.304020
std	38.222625
min	46.000000
25%	76.000000
50%	93.500000

---

75%	125.000000
max	230.000000

```
In [26]: imp = Imputer(missing_values='NaN', strategy='most_frequent', axis=0)
imp.fit(auto2)
auto_frequent = imp.fit_transform(auto2)
auto['horsepower'] = auto_frequent
auto.describe()
```

```
Out[26]:
```

	horsepower
count	398.000000
mean	105.155779
std	38.600986
min	46.000000
25%	76.000000
50%	95.000000
75%	130.000000
max	230.000000

---