## Dimensionality Reduction Techniques

- Here I Covered 4 Method.

1. Missing Value Ratio
2. High Correlation Fillter
3. Low_variance_filter
4. By Random Forest Feature Selection

# 1. Missing Value Ratio (Method- 1)

In [1]:

```
#importing the libraries
import pandas as pd
#reading the file
data = pd.read_csv('missing_value_ratio.csv')
# first 5 rows of the data
data.head()
```

Out[1]:

| | ID | season | holiday | workingday | weather | temp | atemp | humidity | windspeed | count |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | AB101 | 1.0 | 0.0 | 0.0 | 1.0 | 9.84 | 14.395 | 81.0 | NaN | 16 |
| **1** | AB102 | 1.0 | NaN | 0.0 | NaN | 9.02 | 13.635 | 80.0 | NaN | 40 |
| **2** | AB103 | 1.0 | 0.0 | NaN | 1.0 | 9.02 | 13.635 | 80.0 | NaN | 32 |
| **3** | AB104 | NaN | 0.0 | NaN | 1.0 | 9.84 | 14.395 | 75.0 | NaN | 13 |
| **4** | AB105 | 1.0 | NaN | 0.0 | NaN | 9.84 | 14.395 | NaN | 16.9979 | 1 |

See here lot of missing value present in this dataset so here we put missing value ratio.

In [2]:

```
# percentage of missing values in each variable
data.isnull().sum()/len(data)*100
```

Out[2]:

```
ID             0.000000
season         0.069337
holiday       48.497689
workingday     0.069337
weather        0.030817
temp           0.000000
atemp          0.000000
humidity       0.038521
windspeed     41.016949
count          0.000000
dtype: float64
```

In [3]:

```
# saving missing values in a variable
a = data.isnull().sum()/len(data)*100
```

In [4]:
```python
# saving column names in a variable
variables = data.columns
```

In [5]:
```python
# new variable to store variables having missing values less than a threshold
variable = [ ]
for i in range(data.columns.shape[0]):
    if a[i]<=40:   #setting the threshold as 40%
        variable.append(variables[i])
```

- here i remove the columns which has missing value higher than 40%. 40% is just a threshold.
- You set a threshold which has Higher than 40-45 % .

In [6]:
```python
print("--------------------Before remove feature-----------------")
print("Before remove feature: ",variables)
print("--------------------After remove feature-----------------")
print("After remove feature: ",variable)
```

```
--------------------Before remove feature-----------------
Before remove feature:  Index(['ID', 'season', 'holiday', 'workingday', 'weathe
r', 'temp', 'atemp',
       'humidity', 'windspeed', 'count'],
      dtype='object')
--------------------After remove feature-----------------
After remove feature:  ['ID', 'season', 'workingday', 'weather', 'temp', 'atem
p', 'humidity', 'count']
```

Here i remove 'holiday' and 'windspeed' as it has higher missing value ratio

In [7]:
```python
# creating a new dataframe using the above variables
new_data = data[variable]
# first five rows of the new data
new_data.head()
```

Out[7]:

| | ID | season | workingday | weather | temp | atemp | humidity | count |
|---|---|---|---|---|---|---|---|---|
| **0** | AB101 | 1.0 | 0.0 | 1.0 | 9.84 | 14.395 | 81.0 | 16 |
| **1** | AB102 | 1.0 | 0.0 | NaN | 9.02 | 13.635 | 80.0 | 40 |
| **2** | AB103 | 1.0 | NaN | 1.0 | 9.02 | 13.635 | 80.0 | 32 |
| **3** | AB104 | NaN | NaN | 1.0 | 9.84 | 14.395 | 75.0 | 13 |
| **4** | AB105 | 1.0 | 0.0 | NaN | 9.84 | 14.395 | NaN | 1 |

In [8]:
```
# percentage of missing values in each variable of new data
new_data.isnull().sum()/len(new_data)*100
```

Out[8]:
```
ID            0.000000
season        0.069337
workingday    0.069337
weather       0.030817
temp          0.000000
atemp         0.000000
humidity      0.038521
count         0.000000
dtype: float64
```

In [9]:
```
# shape of new and original data
print("Before Remove Mising value RAtio: ",data.shape)
print("Before Remove Mising value RAtio: ",new_data.shape)
```

```
Before Remove Mising value RAtio:  (12980, 10)
Before Remove Mising value RAtio:  (12980, 8)
```

In this way we reduce dimensionality of a data using missing value ratio.

# 2. High Correlation Fillter (Method 2)

- Here I just Demonstrate how you remove high correlation feature.

In [10]:
```
data_me2 = data.copy()
```

In [11]:
```
data_me2 = data_me2.drop(['ID'],axis=1)
```

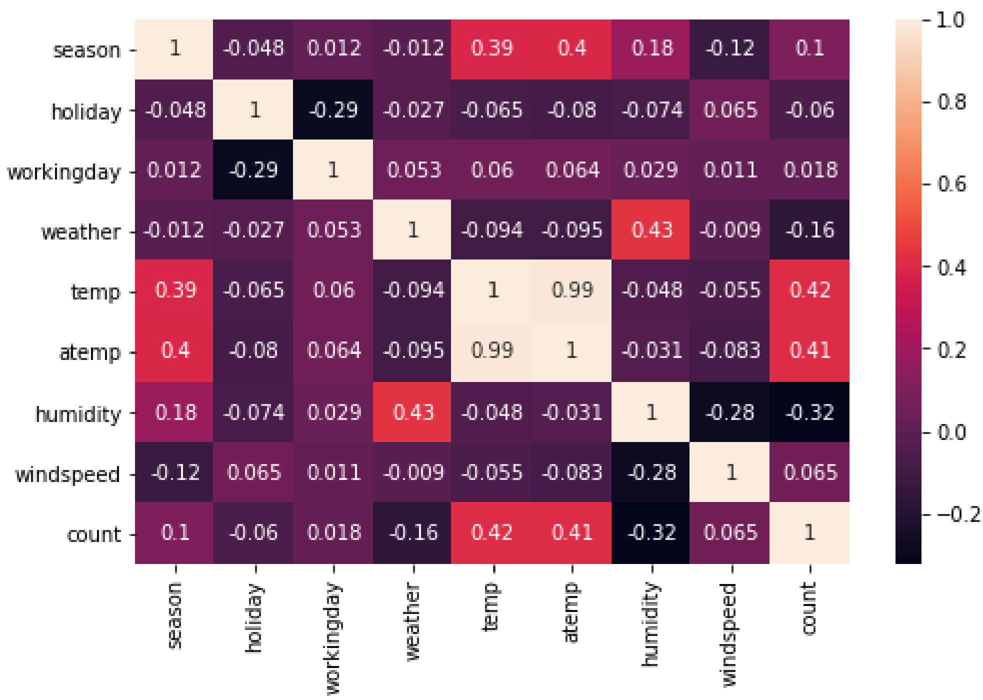
In [12]:
```
correlation = data_me2.corr()
numeric_columns = data_me2.columns

high_corr = [ ]

for c1 in numeric_columns:
  for c2 in numeric_columns:
    if c1 != c2 and c2 not in high_corr and correlation[c1][c2] > 0.9:
      high_corr.append(c1)
```

In [13]:
```
high_corr
```

Out[13]:
```
['temp']
```

In [14]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(8,5))
sns.heatmap(correlation,annot=True)
```

Out[14]:
```
<AxesSubplot:>
```




- See here in heat map atemp and temp have high correlation so my method remove one higher correlation Value.

# 3. Low_variance_filter (Method 3)

In [15]:
```
#importing the libraries
import pandas as pd
from sklearn.preprocessing import normalize
#reading the file
data = pd.read_csv('low_variance_filter.csv')
# first 5 rows of the data
data.head()
```

Out[15]:

| | ID | temp | atemp | humidity | windspeed | count |
|---|---|---|---|---|---|---|
| **0** | AB101 | 9.84 | 14.395 | 81 | 0.0 | 16 |
| **1** | AB102 | 9.02 | 13.635 | 80 | 0.0 | 40 |
| **2** | AB103 | 9.02 | 13.635 | 80 | 0.0 | 32 |
| **3** | AB104 | 9.84 | 14.395 | 75 | 0.0 | 13 |
| **4** | AB105 | 9.84 | 14.395 | 75 | 0.0 | 1 |

In [16]:
```
#percentage of missing values in each variable
data.isnull().sum()/len(data)*100
```

Out[16]:
```
ID           0.0
temp         0.0
atemp        0.0
humidity     0.0
windspeed    0.0
count        0.0
dtype: float64
```

- Here see missing value zero so here we consider variance value to reduce dimensionality.

In [17]:
```
data.var()
```

Out[17]:
```
temp            61.291712
atemp           73.137484
humidity       398.549141
windspeed       69.322053
count        25843.419864
dtype: float64
```

- See here all the variance value shows.
- Always do normalize of your dataset before doing This method.

In [18]:
```
#creating dummy variables of categorical variables
data = data.drop('ID', axis=1)
```

In [19]:
```
normalize = normalize(data)
```

In [20]:
```
data_scaled = pd.DataFrame(normalize)
data_scaled.var()
```

Out[20]:
```
0    0.005877
1    0.007977
2    0.093491
3    0.008756
4    0.111977
dtype: float64
```

- Now use a threshold value to elminate all low var feature.

In [21]:
```
#storing the variance and name of variables
variance = data_scaled.var()
columns = data.columns
```

In [22]:
```
#saving the names of variables having variance more than a threshold value
variable = [ ]
for i in range(0,len(variance)):
    if variance[i]>=0.006:   #setting the threshold as 1%
        variable.append(columns[i])
```

In [23]:
```
print("--------------------Before remove feature-----------------")
print("Before remove feature: ",data.columns)
print("--------------------After remove feature-----------------")
print("After remove feature: ",variable)
```

```
--------------------Before remove feature-----------------
Before remove feature:  Index(['temp', 'atemp', 'humidity', 'windspeed', 'coun
t'], dtype='object')
--------------------After remove feature-----------------
After remove feature:  ['atemp', 'humidity', 'windspeed', 'count']
```

- Here it eliminate temp as it var value is 0.005877.

In [24]:
```
# creating a new dataframe using the above variables
new_data = data[variable]
# first five rows of the new data
new_data.head()
```

Out[24]:

| | atemp | humidity | windspeed | count |
|---|---|---|---|---|
| **0** | 14.395 | 81 | 0.0 | 16 |
| **1** | 13.635 | 80 | 0.0 | 40 |
| **2** | 13.635 | 80 | 0.0 | 32 |
| **3** | 14.395 | 75 | 0.0 | 13 |
| **4** | 14.395 | 75 | 0.0 | 1 |

In [25]:
```python
# shape of new and original data
print("Before Remove Mising value RAtio: ",data.shape)
print("Before Remove Mising value RAtio: ",new_data.shape)
```

```
Before Remove Mising value RAtio:  (12980, 5)
Before Remove Mising value RAtio:  (12980, 4)
```

- In This way we do Low Variance Fillter

# 4. By Random Forest Feature Selection (Method 4)

In [26]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [27]:
```python
from sklearn.datasets import load_iris
import pandas as pd
data = load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Target'] = data.target
df.head()
```

Out[27]:

| | sepal length (cm) | sepal width (cm) | petal length (cm) | petal width (cm) | Target |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | 0 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | 0 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | 0 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | 0 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | 0 |

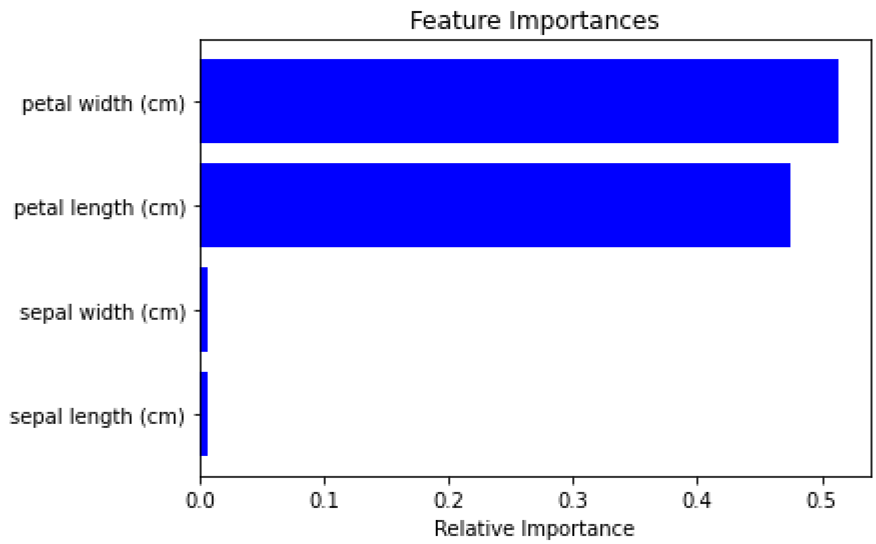
In [28]:
```python
X = df.drop("Target",axis=1)
y = df['Target']
```

In [29]:
```python
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor(random_state=1, max_depth=10)
model.fit(X,y)
```

Out[29]:
```
RandomForestRegressor(max_depth=10, random_state=1)
```

## After fitting the model, plot the feature importance graph:

In [30]:
```
features = df.columns
importances = model.feature_importances_
indices = np.argsort(importances)[-9:]  # top 10 features
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='b', align='center')
plt.yticks(range(len(indices)), [features[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```




- By this you can determine which variable are important which are not . according to this you easily eliminate all less important feature.
- Based on the above graph, we can handpick the top-most features to reduce the dimensionality in our dataset.
- Alternatively, we can use the SelectFromModel of sklearn to do so. It selects the features based on the importance of their weights.

## SelectFromModel of sklearn

In [31]:
```
from sklearn.feature_selection import SelectFromModel
feature = SelectFromModel(model)
X_select = feature.fit_transform(X,y)
```

In [32]:
```
print("--------------------Before remove feature------------------")
print("Before Select: ",X.shape)
print("--------------------After remove feature------------------")
print("After Select: ",X_select.shape)
```

```
--------------------Before remove feature------------------
Before Select:  (150, 4)
--------------------After remove feature------------------
After Select:  (150, 2)
```

In [33]:
```
model.fit(X_select,y)
```

Out[33]:
```
RandomForestRegressor(max_depth=10, random_state=1)
```

- In This way you can do number of feature and easily emliminate this.