CS5805 : Machine Learning I
Lecture #4

Reza Jafari, Ph.D

Collegiate Associate Professor
rjafari@vt.edu

Data Preprocessing

- Data Preprocessing is a broad area and consists of a number of different strategies that are interrelated in complex ways.

Data Preprocessing

- Data Preprocessing is a broad area and consists of a number of different strategies that are interrelated in complex ways.
- Data preprocessing fall into two categories:

Selecting data objects and attributes

Creating/Changing attributes

Data Preprocessing

- Data Preprocessing is a broad area and consists of a number of different strategies that are interrelated in complex ways.
- Data preprocessing fall into two categories:

Selecting data objects and attributes

Creating/Changing attributes

1. Aggregation

Data Preprocessing

- Data Preprocessing is a broad area and consists of a number of different strategies that are interrelated in complex ways.
- Data preprocessing fall into two categories:

Selecting data objects and attributes

Creating/Changing attributes

1. Aggregation
2. Sampling

Data Preprocessing

- Data Preprocessing is a broad area and consists of a number of different strategies that are <u>interrelated</u> in complex ways.
- Data preprocessing fall into two categories:

Selecting data objects and attributes

Creating/Changing attributes

1. Aggregation
2. Sampling
3. Dimensionality reduction

Data Preprocessing

- Data Preprocessing is a broad area and consists of a number of different strategies that are interrelated in complex ways.
- Data preprocessing fall into two categories:

Selecting data objects and attributes

Creating/Changing attributes

1. Aggregation
2. Sampling
3. Dimensionality reduction
4. Feature subset selection

Data Preprocessing

- Data Preprocessing is a broad area and consists of a number of different strategies that are interrelated in complex ways.
- Data preprocessing fall into two categories:

Selecting data objects and attributes

Creating/Changing attributes

1. Aggregation
2. Sampling
3. Dimensionality reduction
4. Feature subset selection
5. Feature selection

Data Preprocessing

- Data Preprocessing is a broad area and consists of a number of different strategies that are interrelated in complex ways.
- Data preprocessing fall into two categories:

Selecting data objects and attributes

Creating/Changing attributes

1. Aggregation
2. Sampling
3. Dimensionality reduction
4. Feature subset selection
5. Feature selection
6. Discretization & binarization

Data Preprocessing

- Data Preprocessing is a broad area and consists of a number of different strategies that are interrelated in complex ways.
- Data preprocessing fall into two categories:

Selecting data objects and attributes

Creating/Changing attributes

1. Aggregation
2. Sampling
3. Dimensionality reduction
4. Feature subset selection
5. Feature selection
6. Discretization & binarization
7. Variable transformation

Data Aggregation

- Sometimes less is more and this is the case with **aggregation**.

Data Aggregation

- Sometimes less is more and this is the case with **aggregation**.
- Aggregation means combining two or more objects into a single object.

Data Aggregation

- Sometimes less is more and this is the case with **aggregation**.
- Aggregation means combining two or more objects into a single object.
- Consider a 'store' data set consisting of item 4 items with the quantity sold during the month M1, M2 and M3.

Data Aggregation

- Sometimes less is more and this is the case with **aggregation**.
- Aggregation means combining two or more objects into a single object.
- Consider a 'store' data set consisting of item 4 items with the quantity sold during the month M1, M2 and M3.
- Three operations : split, apply, combine.

Month	Brand	Quantity
M1	Dove	25
M1	Sunsilk	15
M2	Tide	27
M3	Pantene	44
M2	Sunsilk	12
M3	Dove	8
M2	Pantene	9
M3	Tide	6

Table 1: Aggregated dataset based on the Month-Method : Sum

Month	Quantity
M1	40
M2	48
M3	58

Advantages & Disadvantages

- There are several motivations for aggregation :

Advantages & Disadvantages

- There are several motivations for aggregation :
- First, less memory and processing time. Hence aggregation often enables the use of more expensive data mining algorithm.

Advantages & Disadvantages

- There are several motivations for aggregation :
- First, less memory and processing time. Hence aggregation often enables the use of more expensive data mining algorithm.
- Second, aggregation can act as a change of scope by providing a high-level view of the data instead of low-level view.

Advantages & Disadvantages

- There are several motivations for aggregation :
- First, less memory and processing time. Hence aggregation often enables the use of more expensive data mining algorithm.
- Second, aggregation can act as a change of scope by providing a high-level view of the data instead of low-level view.
- In the precious example aggregation over month gives us a high-level view of sales in M1, M2 and M3.

Advantages & Disadvantages

- There are several motivations for aggregation :
- First, less memory and processing time. Hence aggregation often enables the use of more expensive data mining algorithm.
- Second, aggregation can act as a change of scope by providing a high-level view of the data instead of low-level view.
- In the precious example aggregation over month gives us a high-level view of sales in M1, M2 and M3.
- A disadvantage of aggregation is the potential loss of interesting details.

Data Aggregation - Python

- The aggregation operation split, apply, combine cab be implemented by $\rightarrow$ groupby function with Dataframe in Pandas package.

Data Aggregation - Python

- The aggregation operation split, apply, combine cab be implemented by $\rightarrow$ groupby function with Dataframe in Pandas package.
- Let consider the 'store.csv' data set which contains 6001 objects and 5 attributes.

Data Aggregation - Python

- The aggregation operation split, apply, combine cab be implemented by $\rightarrow$ groupby function with Dataframe in Pandas package.
- Let consider the 'store.csv' data set which contains 6001 objects and 5 attributes.
- Let suppose we are interested in total quantity sold in one month.

Data Aggregation - Python

- The aggregation operation split, apply, combine cab be implemented by $\rightarrow$ groupby function with Dataframe in Pandas package.
- Let consider the 'store.csv' data set which contains 6001 objects and 5 attributes.
- Let suppose we are interested in total quantity sold in one month.
- Or interested in total quantity that is sold by one company.
- Or aggregate based on the 'MBRD' and 'MONTH' with the sum operation.

Data Aggregation - Python

- The aggregation operation split, apply, combine cab be implemented by $\rightarrow$ groupby function with Dataframe in Pandas package.
- Let consider the 'store.csv' data set which contains 6001 objects and 5 attributes.
- Let suppose we are interested in total quantity sold in one month.
- Or interested in total quantity that is sold by one company.
- Or aggregate based on the 'MBRD' and 'MONTH' with the sum operation.
- Or aggregation using multiple operations at once.

Data Aggregation - Python

```python
import pandas as pd
import numpy as np
from IPython.display import display
df = pd.read_csv('store.csv')
df = df.loc[:6000,['MONTH','STORECODE','QTY','VALUE','MBRD']]
# Motive: which feature to group ( split, apply, combine)
# 1. Interested in total quantity sold in one month.
df1 = df.loc[:6000, ['MONTH', 'QTY']]
df2 = df1.groupby('MONTH').sum()
display(df1.head())
display(df2.head())

# 2. Interested in total quantity that is sold by one company.
df3 = df.loc[:6000, ['MBRD', 'QTY']]
df4 = df3.groupby('MBRD').sum()
display(df3.head())
display(df4.head())

# 3. Interested in the quantity of item sold by a company in the respective month
df5 = df.loc[:6000, ['MBRD', 'QTY', 'VALUE','MONTH']]
df6 = df5.groupby(['MBRD', 'MONTH']).sum()
display(df6.head())

# 4. Using aggregate: multiple operations at once
df7 = df.loc[:6000, ['MBRD', 'QTY', 'VALUE','MONTH']]
df8 = df7.groupby(['MBRD', 'MONTH']).agg([np.mean, np.sum, np.std])
display(df8.head())
```

Data Aggregation - Python

- The other approach to implement data aggregation using python is by the window rolling.

Data Aggregation - Python

- The other approach to implement data aggregation using python is by the window rolling.
- The function is called rolling.aggregate(). The windows size needs to be defined per the need.

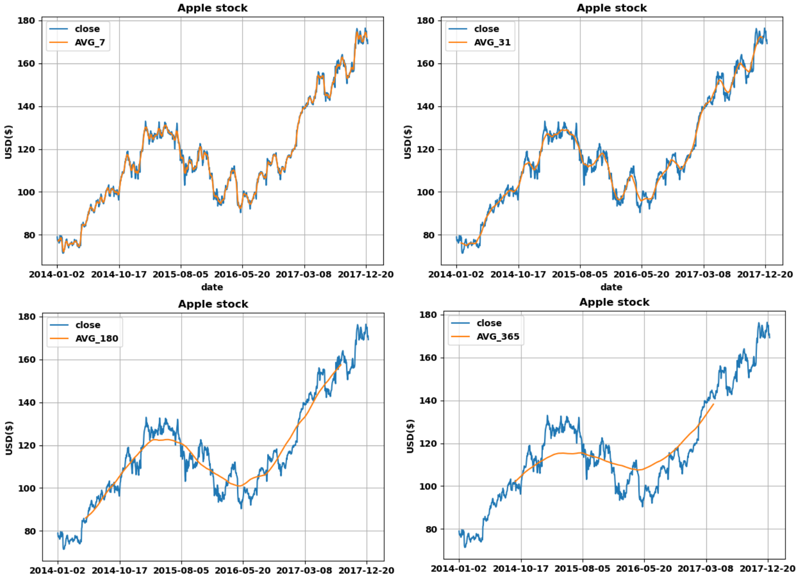
Data Aggregation - Python

- The other approach to implement data aggregation using python is by the window rolling.
- The function is called rolling.aggregate(). The windows size needs to be defined per the need.
- The operation to aggregate also can be defined using the Numpy package.
  - np.sum
  - np.mean
  - np.min
  - np.max
  - np.std

Data Aggregation - Python

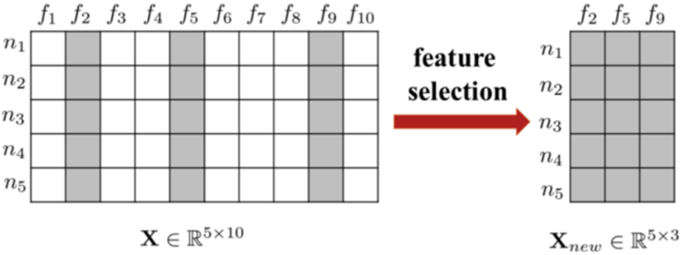
- The other approach to implement data aggregation using python is by the window rolling.
- The function is called rolling.aggregate(). The windows size needs to be defined per the need.
- The operation to aggregate also can be defined using the Numpy package.
  - np.sum
  - np.mean
  - np.min
  - np.max
  - np.std
- The window rolling is a window with variable size that rolls across the data set and perform a specific operation.

Data Aggregation - Window Rolling Approach

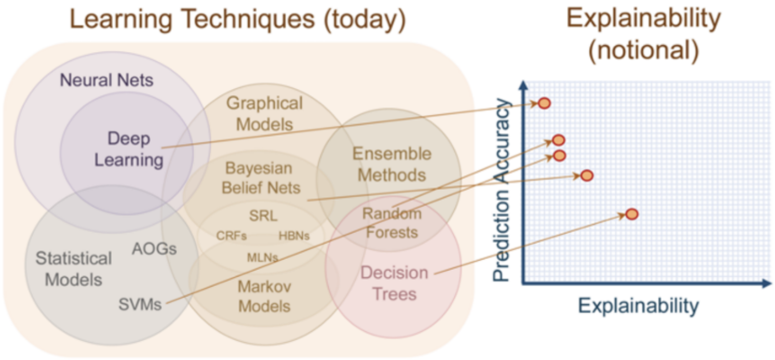


What is feature selection?

- A procedure in machine learning to find a subset of features that produces **better** model for given dataset.
- Avoid overfitting and achieve better generalization ability
- Reduce the storage requirement and training time.
- Interpretability



Interpretability of Learning Algorithms



With feature selection, both the accuracy and interpretability of most learning algorithms can be enhanced

Feature Extraction vs. Feature Selection

Commonalties

- Speed up the learning process
- Reduce the storage requirements
- Build more generalized models

Difference

- Feature extraction obtains new features while feature selection selects a subset of original ones.
- Feature extraction algorithms transform the data onto a new feature space.
- Feature selection maintains physical meanings and gives models better readability and interpretability.

What is Dimensionality Reduction?

- Have you ever worked on a dataset with more than thousands attributes? How about 50000 attributes? It is a very challenging task especially if you don't know where to start.

What is Dimensionality Reduction?

- Have you ever worked on a dataset with more than thousands attributes? How about 50000 attributes? It is a very challenging task especially if you don't know where to start.
- Reducing the dimension of data set to a subset of original dimension is called dimensionality reduction.

What is Dimensionality Reduction?

- Have you ever worked on a dataset with more than thousands attributes? How about 50000 attributes? It is a very challenging task especially if you don't know where to start.
- Reducing the dimension of data set to a subset of original dimension is called dimensionality reduction.
- It is great that we have loads of data for analysis, but it is challenging due to size.

What is Dimensionality Reduction?

- Have you ever worked on a dataset with more than thousands attributes? How about 50000 attributes? It is a very challenging task especially if you don't know where to start.
- Reducing the dimension of data set to a subset of original dimension is called dimensionality reduction.
- It is great that we have loads of data for analysis, but it is challenging due to size.
- We are generating a tremendous amount of data daily. In fact, 90% of the data in the world has been generated in the last 3-4 years!

What is Dimensionality Reduction?

- Have you ever worked on a dataset with more than thousands attributes? How about 50000 attributes? It is a very challenging task especially if you don't know where to start.
- Reducing the dimension of data set to a subset of original dimension is called dimensionality reduction.
- It is great that we have loads of data for analysis, but it is challenging due to size.
- We are generating a tremendous amount of data daily. In fact, 90% of the data in the world has been generated in the last 3-4 years!
  - Facebook collects data of what you like, share, post, places you visit, restaurants you like, etc.

What is Dimensionality Reduction?

- Have you ever worked on a dataset with more than thousands attributes? How about 50000 attributes? It is a very challenging task especially if you don't know where to start.
- Reducing the dimension of data set to a subset of original dimension is called dimensionality reduction.
- It is great that we have loads of data for analysis, but it is challenging due to size.
- We are generating a tremendous amount of data daily. In fact, 90% of the data in the world has been generated in the last 3-4 years!
  - Facebook collects data of what you like, share, post, places you visit, restaurants you like, etc.
  - Your smartphone apps collect a lot of personal information about you

What is Dimensionality Reduction?

- Have you ever worked on a dataset with more than thousands attributes? How about 50000 attributes? It is a very challenging task especially if you don't know where to start.
- Reducing the dimension of data set to a subset of original dimension is called dimensionality reduction.
- It is great that we have loads of data for analysis, but it is challenging due to size.
- We are generating a tremendous amount of data daily. In fact, 90% of the data in the world has been generated in the last 3-4 years!
  - Facebook collects data of what you like, share, post, places you visit, restaurants you like, etc.
  - Your smartphone apps collect a lot of personal information about you
  - Amazon collects data of what you buy, view, click, etc. on their site.

What is Dimensionality Reduction?

- Have you ever worked on a dataset with more than thousands attributes? How about 50000 attributes? It is a very challenging task especially if you don't know where to start.
- Reducing the dimension of data set to a subset of original dimension is called dimensionality reduction.
- It is great that we have loads of data for analysis, but it is challenging due to size.
- We are generating a tremendous amount of data daily. In fact, 90% of the data in the world has been generated in the last 3-4 years!
  - Facebook collects data of what you like, share, post, places you visit, restaurants you like, etc.
  - Your smartphone apps collect a lot of personal information about you
  - Amazon collects data of what you buy, view, click, etc. on their site.
  - Casinos keep a track of every move each customer makes

Why is Dimensionality Reduction required?

- Space required to store the data is reduced as the number of dimensions comes down.

Common dimensionality reduction techniques

Why is Dimensionality Reduction required?

- Space required to store the data is reduced as the number of dimensions comes down.
- Less dimensions lead to less computation/training time.

Common dimensionality reduction techniques

Why is Dimensionality Reduction required?

- Space required to store the data is reduced as the number of dimensions comes down.
- Less dimensions lead to less computation/training time.
- Some algorithms do not perform well when we have a large dimensions. So reducing these dimensions needs to happen for the algorithm to be useful.

Common dimensionality reduction techniques

Why is Dimensionality Reduction required?

- Space required to store the data is reduced as the number of dimensions comes down.
- Less dimensions lead to less computation/training time.
- Some algorithms do not perform well when we have a large dimensions. So reducing these dimensions needs to happen for the algorithm to be useful.
- It takes care of multicollinearity by removing redundant features.

Common dimensionality reduction techniques

Why is Dimensionality Reduction required?

- Space required to store the data is reduced as the number of dimensions comes down.
- Less dimensions lead to less computation/training time.
- Some algorithms do not perform well when we have a large dimensions. So reducing these dimensions needs to happen for the algorithm to be useful.
- It takes care of multicollinearity by removing redundant features.
- It helps in visualizing data.

Common dimensionality reduction techniques

Why is Dimensionality Reduction required?

- Space required to store the data is reduced as the number of dimensions comes down.
- Less dimensions lead to less computation/training time.
- Some algorithms do not perform well when we have a large dimensions. So reducing these dimensions needs to happen for the algorithm to be useful.
- It takes care of multicollinearity by removing redundant features.
- It helps in visualizing data.

Common dimensionality reduction techniques

- Feature selection: Keeps relevant features for learning and removes redundant and irrelevant features

Why is Dimensionality Reduction required?

- Space required to store the data is reduced as the number of dimensions comes down.
- Less dimensions lead to less computation/training time.
- Some algorithms do not perform well when we have a large dimensions. So reducing these dimensions needs to happen for the algorithm to be useful.
- It takes care of multicollinearity by removing redundant features.
- It helps in visualizing data.

Common dimensionality reduction techniques

- Feature selection: Keeps relevant features for learning and removes redundant and irrelevant features
- Dimensionality reduction: Finding a smaller set of new variables, a combination of the input variables, containing basically the same information as the input variable.

Dimensionality Reduction techniques

- Missing Value Ratio.

Dimensionality Reduction techniques

- Missing Value Ratio.
- Low Variance Filter.

Dimensionality Reduction techniques

- Missing Value Ratio.
- Low Variance Filter.
- High Correlation Filter.

Dimensionality Reduction techniques

- Missing Value Ratio.
- Low Variance Filter.
- High Correlation Filter.
- Random Forest

Dimensionality Reduction techniques

- Missing Value Ratio.
- Low Variance Filter.
- High Correlation Filter.
- Random Forest
- Backward Feature elimination

Dimensionality Reduction techniques

- Missing Value Ratio.
- Low Variance Filter.
- High Correlation Filter.
- Random Forest
- Backward Feature elimination
- Forward Feature selection

Dimensionality Reduction techniques

- Missing Value Ratio.
- Low Variance Filter.
- High Correlation Filter.
- Random Forest
- Backward Feature elimination
- Forward Feature selection
- Factor Analysis

Dimensionality Reduction techniques

- Missing Value Ratio.
- Low Variance Filter.
- High Correlation Filter.
- Random Forest
- Backward Feature elimination
- Forward Feature selection
- Factor Analysis
- Principal Component Analysis (PCA)

Dimensionality Reduction techniques

- Missing Value Ratio.
- Low Variance Filter.
- High Correlation Filter.
- Random Forest
- Backward Feature elimination
- Forward Feature selection
- Factor Analysis
- Principal Component Analysis (PCA)
- Independent Component Analysis

Dimensionality Reduction techniques

- Missing Value Ratio.
- Low Variance Filter.
- High Correlation Filter.
- Random Forest
- Backward Feature elimination
- Forward Feature selection
- Factor Analysis
- Principal Component Analysis (PCA)
- Independent Component Analysis
- Methods of Projections

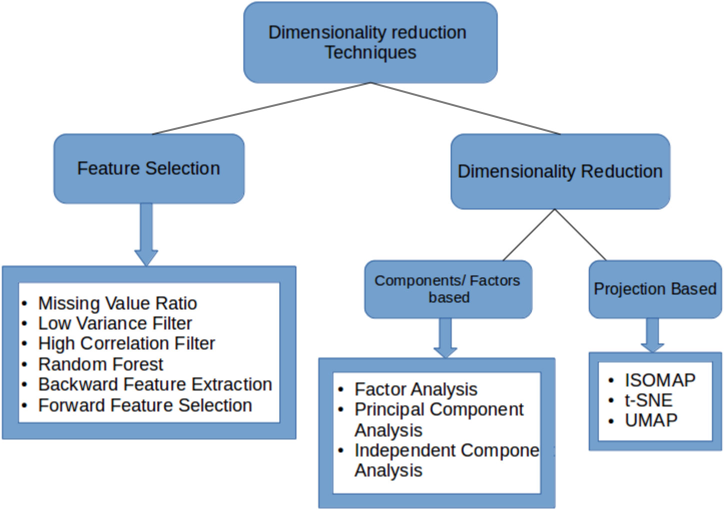
Dimensionality Reduction techniques

- Missing Value Ratio.
- Low Variance Filter.
- High Correlation Filter.
- Random Forest
- Backward Feature elimination
- Forward Feature selection
- Factor Analysis
- Principal Component Analysis (PCA)
- Independent Component Analysis
- Methods of Projections
- t-Distributed Stochastic Neighbor Embedding (t-SNE)

Dimensionality Reduction techniques

- Missing Value Ratio.
- Low Variance Filter.
- High Correlation Filter.
- Random Forest
- Backward Feature elimination
- Forward Feature selection
- Factor Analysis
- Principal Component Analysis (PCA)
- Independent Component Analysis
- Methods of Projections
- t-Distributed Stochastic Neighbor Embedding (t-SNE)
- UMAP

Dimensionality Reduction Technique Summary



Missing Value Ratio

- Given a data set, we would first do EDA.

Missing Value Ratio

- Given a data set, we would first do EDA.
- Let suppose during the EDA, you find that data set has some missing values. Then we may find a method to fill the missing values or impute the missing values.

Missing Value Ratio

- Given a data set, we would first do EDA.
- Let suppose during the EDA, you find that data set has some missing values. Then we may find a method to fill the missing values or impute the missing values.
- What if we have too many missing values? Say more than 50%. Should we impute the missing values or drop the feature?

Missing Value Ratio

- Given a data set, we would first do EDA.
- Let suppose during the EDA, you find that data set has some missing values. Then we may find a method to fill the missing values or impute the missing values.
- What if we have too many missing values? Say more than 50%. Should we impute the missing values or drop the feature?
- We can set a threshold value and if the percentage of missing values in any feature is more than that threshold, we will drop the feature.

Missing Value Ratio Implementation- Python

- Let consider the following data set that has several missing values.

```
# import required libraries
import pandas as pd

threshold = 15
df = pd.read_csv('Train_UWu5bXk.csv')
a = df.isnull().sum()/len(df)*100
print(a)
col = df.columns

variable = []
for i in range(0,df.shape[1]):
    if a[i]<=threshold:
        variable.append(col[i])

df_clean = df[variable]
print(df_clean.isna().sum())
```

Missing Value Ratio Implementation- Python

- Let consider the following data set that has several missing values.
- Original data set contains 12 attributes.

```python
# import required libraries
import pandas as pd

threshold = 15
df = pd.read_csv('Train_UWu5bXk.csv')
a = df.isnull().sum()/len(df)*100
print(a)
col = df.columns

variable = []
for i in range(0,df.shape[1]):
    if a[i]<=threshold:
        variable.append(col[i])

df_clean = df[variable]
print(df_clean.isna().sum())
```

Missing Value Ratio Implementation- Python

- Let consider the following data set that has several missing values.
- Original data set contains 12 attributes.
- Threshold defines as 15%.

```
# import required libraries
import pandas as pd

threshold = 15
df = pd.read_csv('Train_UWu5bXk.csv')
a = df.isnull().sum()/len(df)*100
print(a)
col = df.columns

variable = []
for i in range(0,df.shape[1]):
    if a[i]<=threshold:
        variable.append(col[i])

df_clean = df[variable]
print(df_clean.isna().sum())
```

Missing Value Ratio Implementation- Python

- Let consider the following data set that has several missing values.
- Original data set contains 12 attributes.
- Threshold defines as 15%.
- Remove a feature that has more than the threshold missing values.

```
# import required libraries
import pandas as pd

threshold = 15
df = pd.read_csv('Train_UWu5bXk.csv')
a = df.isnull().sum()/len(df)*100
print(a)
col = df.columns

variable = []
for i in range(0,df.shape[1]):
    if a[i]<=threshold:
        variable.append(col[i])

df_clean = df[variable]
print(df_clean.isna().sum())
```

Low Variance Filter

- Consider a data set where all observations of a feature have the same value, say 1.

Low Variance Filter

- Consider a data set where all observations of a feature have the same value, say 1.
- This feature will not improve the model because it has zero variance.

Low Variance Filter

- Consider a data set where all observations of a feature have the same value, say 1.
- This feature will not improve the model because it has zero variance.
- We need to calculate the variance of each feature and drop the features having low variance as compared to other features in the data set.

Low Variance Filter

- Consider a data set where all observations of a feature have the same value, say 1.
- This feature will not improve the model because it has zero variance.
- We need to calculate the variance of each feature and drop the features having low variance as compared to other features in the data set.
- Feature with low variance will not affect the target variable.

Low Variance Filter - Python

```
# import required libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import normalize
threshold = 15
df = pd.read_csv('Train_UWu5bXk.csv')
a = df.isnull().sum()/len(df)*100

df['Item_Weight'].fillna(df['Item_Weight'].median(), inplace = True)
df['Outlet_Size'].fillna(df['Outlet_Size'].mode()[0], inplace = True)

print(df.isnull().sum()/len(df)*100)

numeric = df.select_dtypes(include= np.number)
normalize = normalize(numeric)
numeric_normalized = pd.DataFrame(normalize)
var = numeric_normalized.var()
var_normalized = numeric_normalized.var()*100/np.max(var)
threshold = 1e-3
variable = []
numeric_col = numeric_normalized.columns
for i in range(len(numeric_col)):
    if var_normalized[i] < threshold:
        variable.append(i)

numeric_clean = numeric.drop(numeric.columns[variable], axis=1)
```

High Correlation Filter

- High correlation between two variables means they have linearly similar and are likely to carry similar information.

High Correlation Filter

- High correlation between two variables means they have linearly similar and are likely to carry similar information.
- We can calculate the correlation between independent features that are numerical in nature.

High Correlation Filter

- High correlation between two variables means they have linearly similar and are likely to carry similar information.
- We can calculate the correlation between independent features that are numerical in nature.
- If the correlation coefficient crosses a certain threshold value, we can drop on the features to avoid the collinearity issue in the regression anlysis.

High Correlation Filter

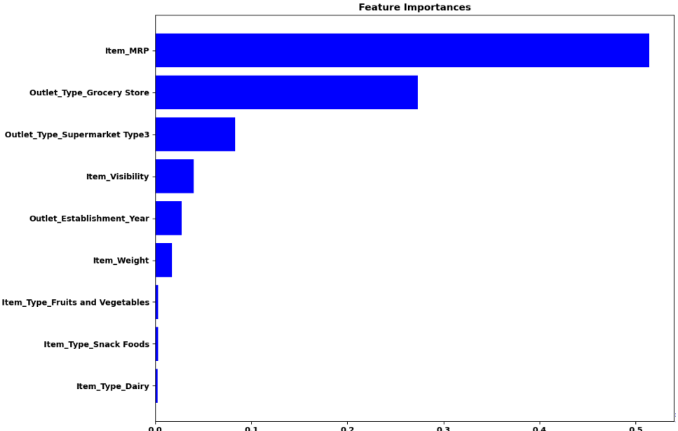
- High correlation between two variables means they have linearly similar and are likely to carry similar information.
- We can calculate the correlation between independent features that are numerical in nature.
- If the correlation coefficient crosses a certain threshold value, we can drop on the features to avoid the collinearity issue in the regression anlysis.
- E.g., data set containing two features : left foot size and right foot size with the target variable : height.

High Correlation Filter

- High correlation between two variables means they have linearly similar and are likely to carry similar information.
- We can calculate the correlation between independent features that are numerical in nature.
- If the correlation coefficient crosses a certain threshold value, we can drop on the features to avoid the collinearity issue in the regression anlysis.
- E.g., data set containing two features : left foot size and right foot size with the target variable : height.
- Dataframe.corr() can be used in python to calculate the Pearson correlation coefficient between all numerical features.

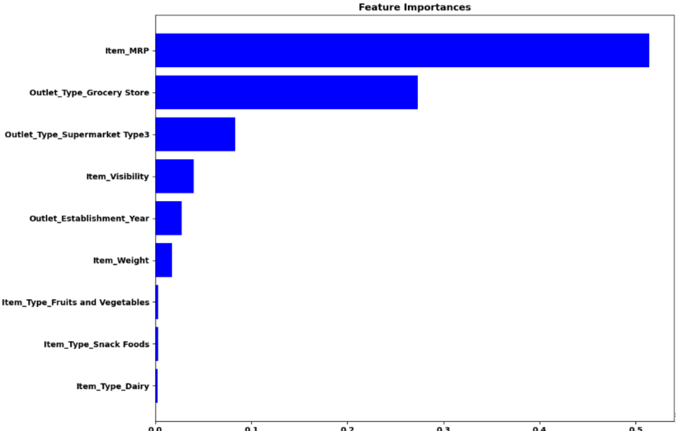
Random Forest

- Random Forest is one of the most widely used algorithms for feature selection.



Random Forest

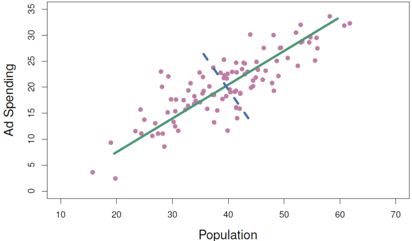
- Random Forest is one of the most widely used algorithms for feature selection.
- It comes packaged with in-built feature importance so need to be programmed separately.



Random Forest - Python

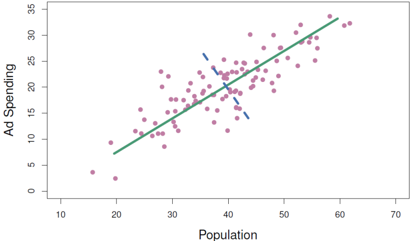
```
import numpy as np
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt

train = pd.read_csv('Train_UWu5bXk.csv')
df = train.drop('Item_Outlet_Sales', 1)
df = df.drop(['Item_Identifier', 'Outlet_Identifier'], axis=1)
df['Item_Weight'].fillna(df['Item_Weight'].median(), inplace = True)
df['Outlet_Size'].fillna(df['Outlet_Size'].mode()[0], inplace = True)
model = RandomForestRegressor(random_state=1, max_depth=10)
df=pd.get_dummies(df)
model.fit(df,train.Item_Outlet_Sales)
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.tight_layout()
plt.show()
```

- Principal Component Analysis (PCA) is a popular approach for deriving a low-dimensional set of features from a large set of variables.



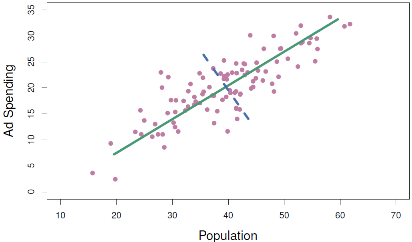
Principal Component Analysis

- Principal Component Analysis (PCA) is a popular approach for deriving a low-dimensional set of features from a large set of variables.
- PCA is a technique for reducing the dimension of an $n \times p$ data matrix X.



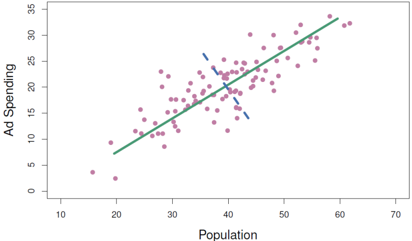
Principal Component Analysis

- Principal Component Analysis (PCA) is a popular approach for deriving a low-dimensional set of features from a large set of variables.
- PCA is a technique for reducing the dimension of an $n \times p$ data matrix X.
- The first principal component direction of the data along which the observations vary the most.



Principal Component Analysis

- Principal Component Analysis (PCA) is a popular approach for deriving a low-dimensional set of features from a large set of variables.
- PCA is a technique for reducing the dimension of an $n \times p$ data matrix X.
- The first principal component direction of the data along which the observations vary the most.
- *Projected observations* on the green solid line (first principal), gives the largest possible variance.



Principal Component Analysis-Python

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

train = pd.read_csv('Train_UWu5bXk.csv')
df = train.drop('Item_Outlet_Sales', axis = 1)
df = df.drop(['Item_Identifier', 'Outlet_Identifier'], axis=1)
df['Item_Weight'].fillna(df['Item_Weight'].median(), inplace = True)
df['Outlet_Size'].fillna(df['Outlet_Size'].mode()[0], inplace = True)

scaler = MinMaxScaler()
X = scaler.fit_transform(df.select_dtypes(include = np.number))

pca = PCA(n_components = 'mle',svd_solver='full')
pca.fit(X)
X_PCA = pca.transform(X)
print("Original Dim", X.shape)
print("Transformed Dim", X_PCA.shape)
PCA(n_components='mle',svd_solver='full')
print(f'explained variance ratio {pca.explained_variance_ratio_}')
print(f'singular values of transformed datam{pca.singular_values_}')
plt.plot(np.arange(1, len(np.cumsum(pca.explained_variance_ratio_))+1, 1),
         np.cumsum(pca.explained_variance_ratio_))
plt.xticks(np.arange(1, len(np.cumsum(pca.explained_variance_ratio_))+1, 1))
plt.grid()
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance')
plt.show()
```

Discretization

- Converting numerical $\rightarrow$ categorical features is called Discretization.

Discretization

- Converting numerical $\rightarrow$ categorical features is called Discretization.
- Let suppose a data set contains an 'Age' as a numerical feature and we want to convert the numerical feature 'Age' into categorical ordinal feature $\{$Young,Mature, Old $\}$.

Age	10, 11, 13, 32, 34, 40, 72, 73, 75

Discretization

- Converting numerical $\rightarrow$ categorical features is called Discretization.
- Let suppose a data set contains an 'Age' as a numerical feature and we want to convert the numerical feature 'Age' into categorical ordinal feature {Young,Mature, Old }.

Age	10, 11, 13, 32, 34, 40, 72, 73, 75

- One method is to define 3 bins or buckets and make the conversion.

Attribute	Age	Age	Age
	10, 11, 13	32, 34, 40	72, 73, 75
After Discretization	Young	Mature	Old

Binning - Equal width

- One method of discretization is through binning approach.

Binning - Equal width

- One method of discretization is through binning approach.
- There are two types of binning approaches:

Equal width (or distance)

Binning - Equal width

- One method of discretization is through binning approach.
- There are two types of binning approaches:

Equal width (or distance)

- The simplest binning approach is to partition the range of the variable into k equal-width intervals.

Binning - Equal width

- One method of discretization is through binning approach.
- There are two types of binning approaches:

Equal width (or distance)

- The simplest binning approach is to partition the range of the variable into k equal-width intervals.
- The interval width is simply the range [min, max] of the variable divided by k.

$$w = \frac{max - min}{k}$$

Binning - Equal width

- One method of discretization is through binning approach.
- There are two types of binning approaches:

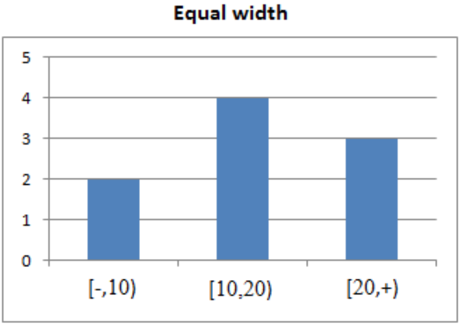
Equal width (or distance)

- The simplest binning approach is to partition the range of the variable into k equal-width intervals.
- The interval width is simply the range [min, max] of the variable divided by k.

$$w = \frac{max - min}{k}$$

- i^{th} interval range will be $[\ min + (i-1)w, min + iw]$ where $i = 1, 2, ..., k$

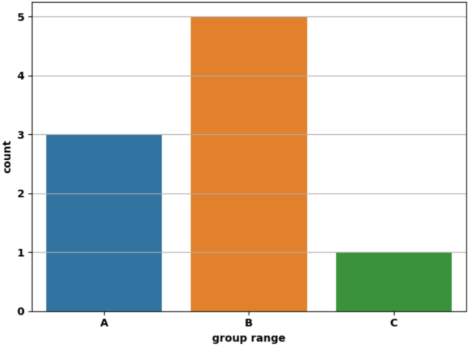
Binning - Equal width



Equal width binning-Python

- The pd.cut() inside pandas package can be used to implement the binning discretization using the equal width approach.

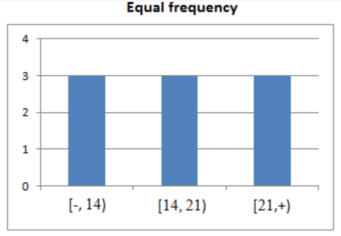
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
number = np.array([1,4,12,16,16,18,17,2,28])


df = pd.DataFrame(data=number, columns=['Number'])
df['group'] = pd.cut(df['Number'], bins = 3
                    ,labels=['A','B','C'])
df.group = df.group.astype('object')
plt.figure()
sns.countplot(x = 'group', data = df)
plt.xlabel('group range')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



Binning- Equal frequency

Equal frequency

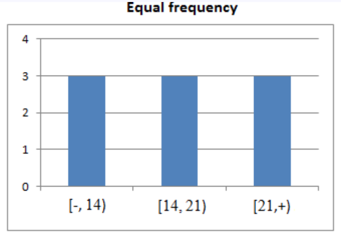
- Unlike equal width, equal frequency binning does not have equal width in each bin.



Binning- Equal frequency

Equal frequency

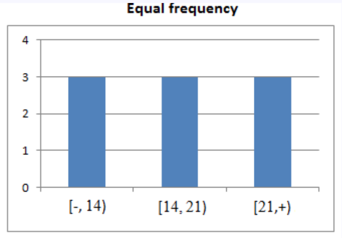
- Unlike equal width, equal frequency binning does not have equal width in each bin.
- However each bin contains an equal amount of observations.



Binning- Equal frequency

Equal frequency

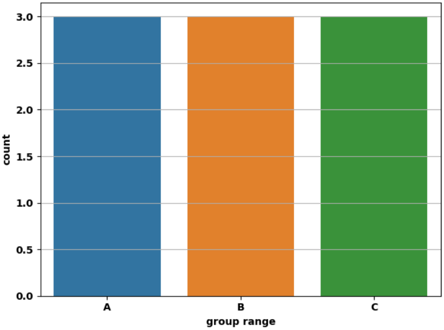
- Unlike equal width, equal frequency binning does not have equal width in each bin.
- However each bin contains an equal amount of observations.
- In the Equal frequency discretization the threshold of all bins is selected in a way that all bins contain the same number of numerical values.



Equal frequency binning-Python

- The pd.qcut() inside pandas package can be used to implement the binning discretization using the equal frequency approach.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns

number = np.array([1,4,12,16,16,18,17,2,28])

df = pd.DataFrame(data=number, columns=['Number'])
df['group'] = pd.qcut(df['Number'], q=3, precision=0
                      ,labels=['A','B','C'])
df.group = df.group.astype('object')

plt.figure()
sns.countplot(x = 'group', data = df)
plt.xlabel('group range')
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```



Encoding Categorical values

- In many practical Data Science activities, the data set will contain categorical variables.

Encoding Categorical values

- In many practical Data Science activities, the data set will contain categorical variables.
  - color : {red, yellow, blue}

Encoding Categorical values

- In many practical Data Science activities, the data set will contain categorical variables.
  - color : {red, yellow, blue}
  - size : {samll, medium, large}

Encoding Categorical values

- In many practical Data Science activities, the data set will contain categorical variables.
    - color : {red, yellow, blue}
    - size : {samll, medium, large}
    - Geographical designations: {state, country}

Encoding Categorical values

- In many practical Data Science activities, the data set will contain categorical variables.
  - color : {red, yellow, blue}
  - size : {samll, medium, large}
  - Geographical designations: {state, country}
- The challenge is determining how to use categorical data in the analysis while many machine learning algorithms are NOT supporting categorical data.

Encoding Categorical values

- In many practical Data Science activities, the data set will contain categorical variables.
  - color : {red, yellow, blue}
  - size : {samll, medium, large}
  - Geographical designations: {state, country}
- The challenge is determining how to use categorical data in the analysis while many machine learning algorithms are NOT supporting categorical data.
- As with many other aspects of the Data Science world, there is no single answer to approach this problem.

Encoding Categorical values

- In many practical Data Science activities, the data set will contain categorical variables.
  - color : {red, yellow, blue}
  - size : {samll, medium, large}
  - Geographical designations: {state, country}
- The challenge is determining how to use categorical data in the analysis while many machine learning algorithms are NOT supporting categorical data.
- As with many other aspects of the Data Science world, there is no single answer to approach this problem.
- Fortunately, the python tools of pandas and scikit-learn provide several approaches to transform the categorical data $\rightarrow$ numeric values.

Encoding Categorical values using Find and Replace

- We are going to cover three approaches:

1- Find and Replace

```
cleanup_nums = {"num_doors":     {"four": 4, "two": 2},
                "num_cylinders": {"four": 4, "six": 6, "five": 5, "eight": 8,
                                  "two": 2, "twelve": 12, "three":3 }}
```

Encoding Categorical values using Find and Replace

- We are going to cover three approaches:

1- Find and Replace

- E.g., 'two'$\rightarrow$ 2 or 'four'$\rightarrow$ 4

```
cleanup_nums = {"num_doors":     {"four": 4, "two": 2},
                "num_cylinders": {"four": 4, "six": 6, "five": 5, "eight": 8,
                                  "two": 2, "twelve": 12, "three":3 }}
```

Encoding Categorical values using Find and Replace

- We are going to cover three approaches:

1- Find and Replace

- E.g., 'two'$\rightarrow$ 2 or 'four'$\rightarrow$ 4
- Find all fours (categorical) in the feature column and replace by numeric 4.

```
cleanup_nums = {"num_doors":     {"four": 4, "two": 2},
                "num_cylinders": {"four": 4, "six": 6, "five": 5, "eight": 8,
                                  "two": 2, "twelve": 12, "three":3 }}
```

Encoding Categorical values using Find and Replace

- We are going to cover three approaches:

1- Find and Replace

- E.g., 'two'$\rightarrow$ 2 or 'four'$\rightarrow$ 4
- Find all fours (categorical) in the feature column and replace by numeric 4.
- Pandas makes it easy for us to directly replace the text values with their numeric equivalent using replace.

```
cleanup_nums = {"num_doors":     {"four": 4, "two": 2},
                "num_cylinders": {"four": 4, "six": 6, "five": 5, "eight": 8,
                                  "two": 2, "twelve": 12, "three":3 }}
```

Encoding Categorical values using Find and Replace

- We are going to cover three approaches:

1- Find and Replace

- E.g., 'two'$\rightarrow$ 2 or 'four'$\rightarrow$ 4
- Find all fours (categorical) in the feature column and replace by numeric 4.
- Pandas makes it easy for us to directly replace the text values with their numeric equivalent using replace.
- In the Auto dataset two categorical features are: 'num-doors' and 'num-cylinders' with values 'two' & 'four' $\rightarrow$ num-doors and 'two', 'four', 'five', 'six', 'eight' & 'twelve' $\rightarrow$ num-cylinders.

```
cleanup_nums = {"num_doors":     {"four": 4, "two": 2},
                "num_cylinders": {"four": 4, "six": 6, "five": 5, "eight": 8,
                                  "two": 2, "twelve": 12, "three":3 }}
```

Encoding Categorical values using Label Encoding

- Another to encoding categorical values is to use a technique called label encoding.

2- Label Encoding

Encoding Categorical values using Label Encoding

- Another to encoding categorical values is to use a technique called label encoding.

2- Label Encoding

- Label encoding is simply converting each value in a column to a number.

Encoding Categorical values using Label Encoding

- Another to encoding categorical values is to use a technique called label encoding.

2- Label Encoding

- Label encoding is simply converting each value in a column to a number.
- In the Auto dataset, the body-style contains 5 different values. We could choose to encode it like this:

Encoding Categorical values using Label Encoding

- Another to encoding categorical values is to use a technique called label encoding.

2- Label Encoding

- Label encoding is simply converting each value in a column to a number.
- In the Auto dataset, the body-style contains 5 different values. We could choose to encode it like this:
  - convertible $\rightarrow$ 0

Encoding Categorical values using Label Encoding

- Another to encoding categorical values is to use a technique called label encoding.

2- Label Encoding

- Label encoding is simply converting each value in a column to a number.
- In the Auto dataset, the body-style contains 5 different values. We could choose to encode it like this:
  - convertible $\rightarrow$ 0
  - hardtop $\rightarrow$ 1

Encoding Categorical values using Label Encoding

- Another to encoding categorical values is to use a technique called label encoding.

2- Label Encoding

- Label encoding is simply converting each value in a column to a number.
- In the Auto dataset, the body-style contains 5 different values. We could choose to encode it like this:
  - convertible $\rightarrow$ 0
  - hardtop $\rightarrow$ 1
  - hatchback $\rightarrow$ 2

Encoding Categorical values using Label Encoding

- Another to encoding categorical values is to use a technique called label encoding.

2- Label Encoding

- Label encoding is simply converting each value in a column to a number.
- In the Auto dataset, the body-style contains 5 different values. We could choose to encode it like this:
  - convertible $\rightarrow$ 0
  - hardtop $\rightarrow$ 1
  - hatchback $\rightarrow$ 2
  - sedan $\rightarrow$ 3

Encoding Categorical values using Label Encoding

- Another to encoding categorical values is to use a technique called label encoding.

2- Label Encoding

- Label encoding is simply converting each value in a column to a number.
- In the Auto dataset, the body-style contains 5 different values. We could choose to encode it like this:
  - convertible $\rightarrow$ 0
  - hardtop $\rightarrow$ 1
  - hatchback $\rightarrow$ 2
  - sedan $\rightarrow$ 3
  - wagon $\rightarrow$ 4

Encoding Categorical values- Python

- Label encoding can be implemented using the .cat.codes function inside the pandas package.

```
#-----------------------------
# Approach 2 - Label Encoding
#-----------------------------
df['body_style'] = df['body_style'].astype('category')
df["body_style_cat"] = df["body_style"].cat.codes
#-----------------------------
# Manual Label Encoding
#-----------------------------
df["body_style_cat-manual"] = np.where(df["body_style"]=='convertible',0,
                                       np.where(df["body_style"]=='hardtop',1,
                                       np.where(df["body_style"]=='hatchback',2,
                                       np.where(df["body_style"] == 'sedan', 3,
                                       np.where(df["body_style"]=='wagon',4,''))
                                      )))
comp = df[['body_style_cat','body_style_cat-manual']]
```

Encoding Categorical values- Python

- Label encoding can be implemented using the .cat.codes function inside the pandas package.
- Alternatively it can be implemented manually using np.where function inside the numpy package.

```
#-----------------------------
# Approach 2 - Label Encoding
#----------------------------
df['body_style'] = df['body_style'].astype('category')
df["body_style_cat"] = df["body_style"].cat.codes
#------------------------------
# Manual Label Encoding
#----------------------------
df["body_style_cat-manual"] = np.where(df["body_style"]=='convertible',0,
                                       np.where(df["body_style"]=='hardtop',1,
                                       np.where(df["body_style"]=='hatchback',2,
                                       np.where(df["body_style"] == 'sedan', 3,
                                       np.where(df["body_style"]=='wagon',4,''))
                                      )))
comp = df[['body_style_cat','body_style_cat-manual']]
```

Encoding Categorical values using One Hot Encoding

- Label encoding has the advantage that it is straightforward.

3 - One Hot Encoding

Encoding Categorical values using One Hot Encoding

- Label encoding has the advantage that it is straightforward.
- But it has disadvantage that the numeric values can be misinterpreted by the algorithms.

3 - One Hot Encoding

Encoding Categorical values using One Hot Encoding

- Label encoding has the advantage that it is straightforward.
- But it has disadvantage that the numeric values can be misinterpreted by the algorithms.
- For example, the value of 0 is obviously less than the value of 4 but does that really correspond to the data set in real life? Does a wagon have '4X' more weight in our calculation than the convertible?

3 - One Hot Encoding

- A common alternative approach is called One Hot Encoding.
- The basic strategy is to convert each category value into a new column and assigns 1 or 0 (true or false) value to the column.

Encoding Categorical values using One Hot Encoding

- Label encoding has the advantage that it is straightforward.
- But it has disadvantage that the numeric values can be misinterpreted by the algorithms.
- For example, the value of 0 is obviously less than the value of 4 but does that really correspond to the data set in real life? Does a wagon have '4X' more weight in our calculation than the convertible?

3 - One Hot Encoding

- A common alternative approach is called One Hot Encoding.
- The basic strategy is to convert each category value into a new column and assigns 1 or 0 (true or false) value to the column.
- This has the benefit of not weighting a value improperly but does have the downside of adding more columns to the data set.

One Hot Encoding- Python

- Pandas support this feature using get-dummies. The function is named this way because it creates dummy indicator variable (0 1).

```
#----------------------------
# Approach 3 - One Hot Encoding
#----------------------------
dum_original = df['drive_wheels']
df = pd.get_dummies(df, columns=["drive_wheels"])
df = pd.concat([df, dum_original], axis=1)
df.head()
df_with_dummy_com = df[['drive_wheels','drive_wheels_4wd',
       'drive_wheels_fwd', 'drive_wheels_rwd']]
print(df_with_dummy_com.to_string())
```

One Hot Encoding- Python

- Pandas support this feature using get-dummies. The function is named this way because it creates dummy indicator variable (0 1).
- In the Auto dataset the feature drive-wheels have values of '4w', 'fwd', 'rwd'.

```
#-----------------------------
# Approach 3 - One Hot Encoding
#-----------------------------
dum_original = df['drive_wheels']
df = pd.get_dummies(df, columns=["drive_wheels"])
df = pd.concat([df, dum_original], axis=1)
df.head()
df_with_dummy_com = df[['drive_wheels','drive_wheels_4wd',
       'drive_wheels_fwd', 'drive_wheels_rwd']]
print(df_with_dummy_com.to_string())
```

One Hot Encoding- Python

- Pandas support this feature using get-dummies. The function is named this way because it creates dummy indicator variable (0 1).
- In the Auto dataset the feature drive-wheels have values of '4w', 'fwd', 'rwd'.
- By using get-dummies we can convert this in to three columns with 0 and 1.

```
#----------------------------
# Approach 3 - One Hot Encoding
#---------------------------
dum_original = df['drive_wheels']
df = pd.get_dummies(df, columns=["drive_wheels"])
df = pd.concat([df, dum_original], axis=1)
df.head()
df_with_dummy_com = df[['drive_wheels','drive_wheels_4wd',
       'drive_wheels_fwd', 'drive_wheels_rwd']]
print(df_with_dummy_com.to_string())
```