Dimensionality Reduction

Dr. Venkataramana Veeramsetty

June 24, 2020

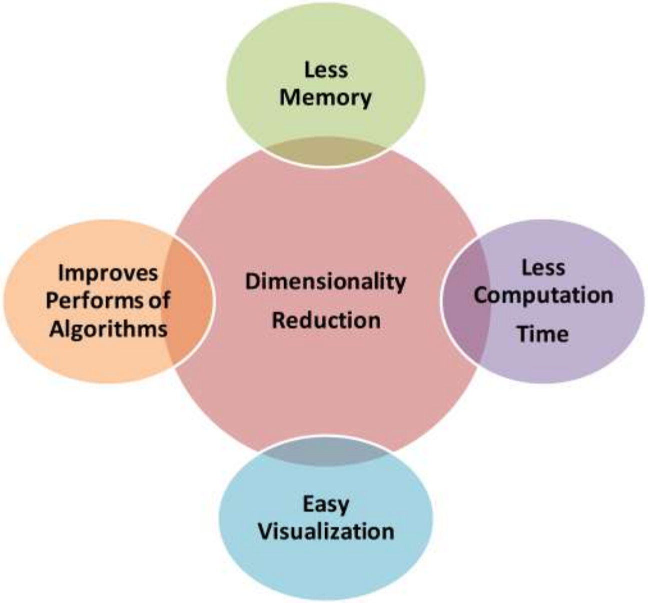
Data Generation

As data generation and collection keeps increasing, visualizing it and drawing inferences becomes more and more challenging.

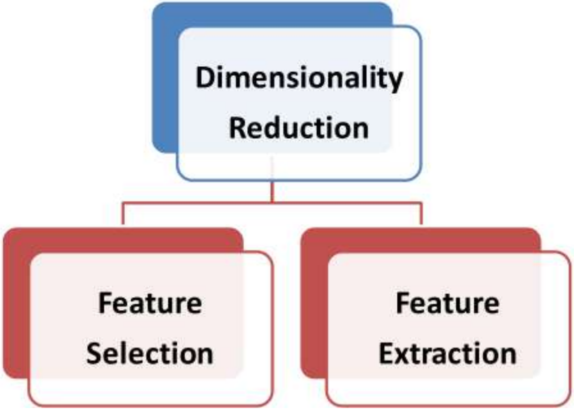
What is Dimensionality Reduction

- √ Dimensionality Reduction is the concept to reduce the number of features in the dataset without having to lose much information and keep (or improve) the model's performance.
- √ It's a really powerful way to deal with huge datasets

Why Dimensionality Reduction

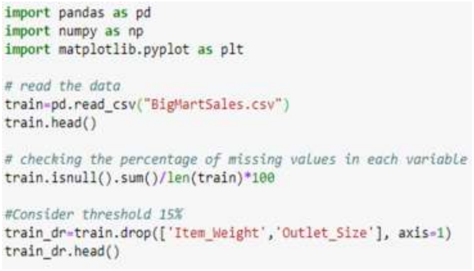


Dimensionality Reduction Classification



Missing Value Ratio

- √ If percentage of missing values for an input features more than threshold then drop that feature, will reduce dimension of the dataset

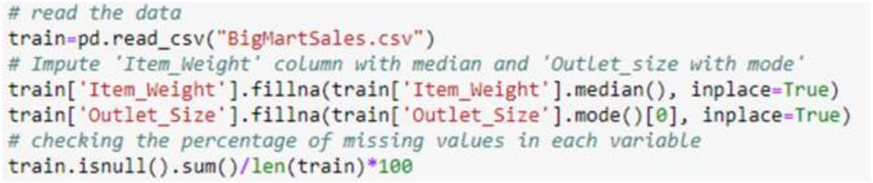


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# read the data
train=pd.read_csv("BigMartSales.csv")
train.head()

# checking the percentage of missing values in each variable
train.isnull().sum()/len(train)*100

#Consider threshold 15%
train_dr=train.drop(['Item_Weight','Outlet_Size'], axis=1)
train_dr.head()
```

- √ If percentage of missing values is low then we can impute but it is more it is better to drop

Imputation

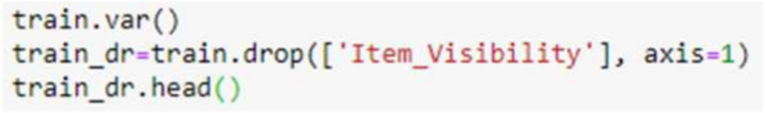
√ For future use of "BigMartSales" Dataset, try to impute missing values with "Median" of the feature



```
# read the data
train=pd.read_csv("BigMartSales.csv")
# Impute 'Item_Weight' column with median and 'Outlet_size with mode'
train['Item_Weight'].fillna(train['Item_Weight'].median(), inplace=True)
train['Outlet_Size'].fillna(train['Outlet_Size'].mode()[0], inplace=True)
# checking the percentage of missing values in each variable
train.isnull().sum()/len(train)*100
```

Low Variance Filter

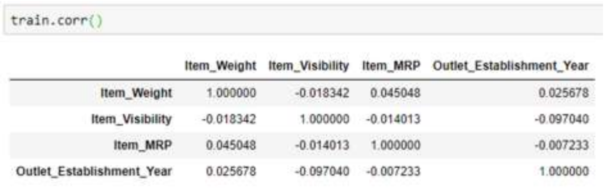
- √ Machine will not generalized or not learn properly from the features having low variance
- √ Drop features having low variance from the dataset



```
train.var()
train_dr=train.drop(['Item_Visibility'], axis=1)
train_dr.head()
```

High Correlation filter

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



```
train.corr()
```

	Item_Weight	Item_Visibility	Item_MRP	Outlet_Establishment_Year
Item_Weight	1.000000	-0.018342	0.045048	0.025678
Item_Visibility	-0.018342	1.000000	-0.014013	-0.097040
Item_MRP	0.045048	-0.014013	1.000000	-0.007233
Outlet_Establishment_Year	0.025678	-0.097040	-0.007233	1.000000

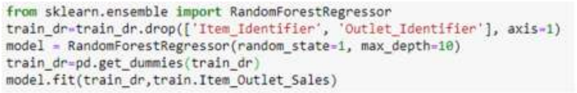
Wonderful, we don't have any variables with a high correlation in our dataset. Generally, if the correlation between a pair of variables is greater than 0.5-0.6, we should seriously consider dropping one of those variables.

Random Forest

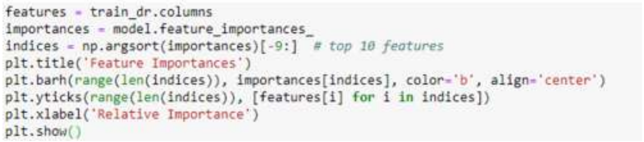
Random Forest is one of the most widely used algorithms for feature selection. It comes packaged with in-built feature importance so you don't need to program that separately. This helps us select a smaller subset of features.

Random Forest

- √ Random Forest is one of the most widely used algorithms for feature selection.
- √ We need to convert the data into numeric form by applying one hot encoding
- √ Let's also drop the ID variables ($Item_Identifier$ and $Outlet_Identifier$) as these are just unique numbers and hold no significant importance for us currently.

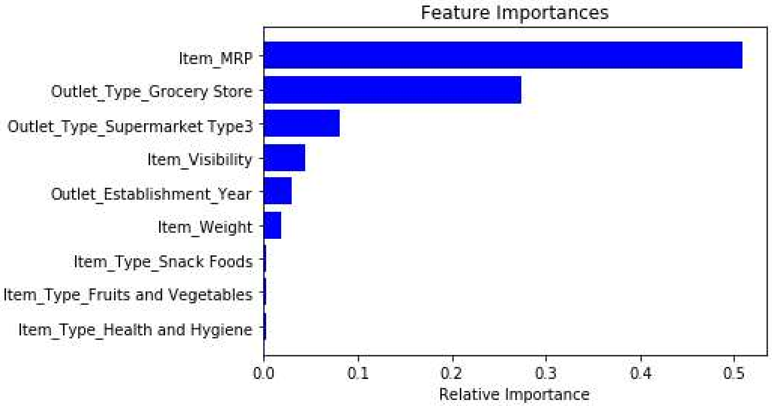


```python
from sklearn.ensemble import RandomForestRegressor
train_dr=train_dr.drop(['Item_Identifier', 'Outlet_Identifier'], axis=1)
model = RandomForestRegressor(random_state=1, max_depth=10)
train_dr=pd.get_dummies(train_dr)
model.fit(train_dr,train.Item_Outlet_Sales)
```



```python
features = train_dr.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()
```

Feature Importance Graph



Based on the above graph, we can hand pick the top-most features to reduce the dimensionality in our dataset.

Forward Feature Selection

Step-1 Read number of features (n), SL=0.05

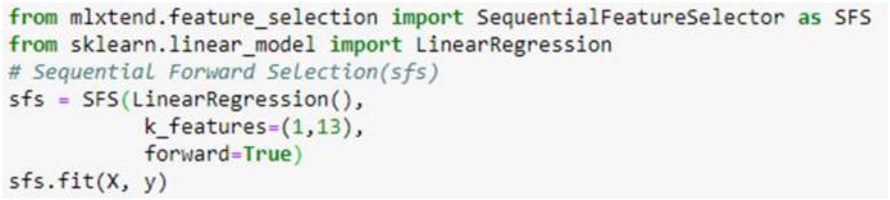
Step-2 Train "n" models each has one input feature

Step-3 Pick the model with lowest p-value, if p-value less than SL then append that feature to dataset

Step-4 Add remaining features one by one to above selected feature and run the model.

Step-5 Repeat step-3, if lowest p-value not less than SL terminate the process, choose previous previous feature combination

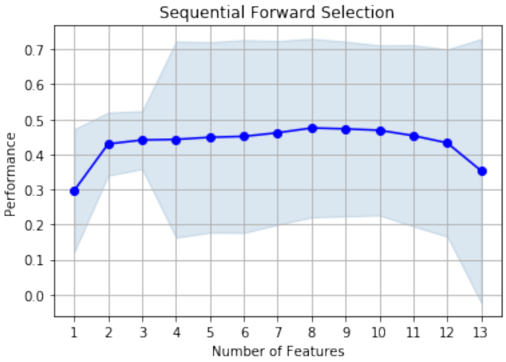
Python code



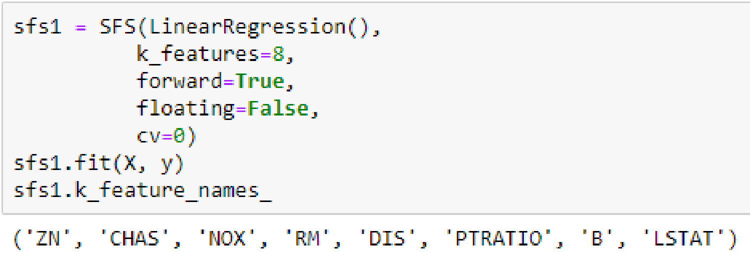
```python
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from sklearn.linear_model import LinearRegression
# Sequential Forward Selection(sfs)
sfs = SFS(LinearRegression(),
          k_features=(1,13),
          forward=True)
sfs.fit(X, y)
```

Python code

```
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs
import matplotlib.pyplot as plt
fig1 = plot_sfs(sfs.get_metric_dict(), kind='std_dev')
plt.title('Sequential Forward Selection')
plt.grid()
plt.show()
```



Python code



```
sfs1 = SFS(LinearRegression(),
           k_features=8,
           forward=True,
           floating=False,
           cv=0)
sfs1.fit(X, y)
sfs1.k_feature_names_
```

```
('ZN', 'CHAS', 'NOX', 'RM', 'DIS', 'PTRATIO', 'B', 'LSTAT')
```

Backward Feature Elimination

step-1 Choose a significance level SL=0.05

step-2 Fit a full model including all the features

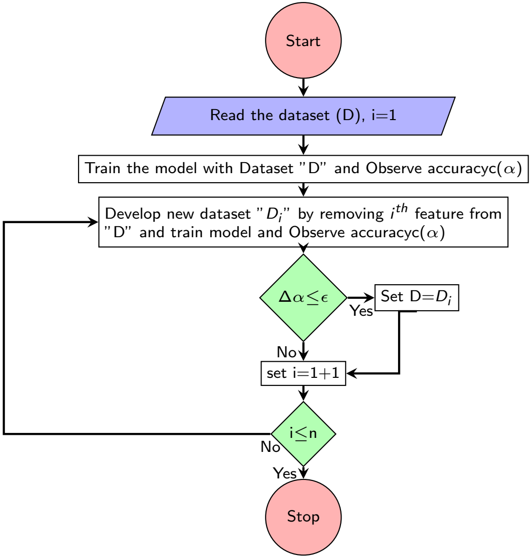
step-3 Find feature corresponds to highest p-value

step-4 If p-value>SL goto step-5 else break

step-5 Remove that feature from Dataset and goto step-2

Use same python code by keeping forward="False"

Backward Feature Elimination



Start
Read the dataset (D), i=1
Train the model with Dataset "D" and Observe accuracyc(α)
Develop new dataset "D_i" by removing i^{th} feature from "D" and train model and Observe accuracyc(α)
$\Delta\alpha \leq \epsilon$
Yes
Set D=D_i
No
set i=1+1
i≤n
No
Yes
Stop

Use same python code by keeping forward="False"

Bi-directional elimination

- √ It is combination of Forward Feature Selection (FFS) and Backward Feature Elimination (BFE)
- √ It is similar to forward selection but the difference is while adding a new feature it also checks the significance of already added features and if it finds any of the already selected features insignificant then it simply removes that particular feature through backward elimination.

step-1 Choose a significance level to enter and exit the model (e.g. $SL_{in} = 0.05$ and $SL_{out} = 0.05$ with 95

step-2 Train "n" models each has one input feature

Step-3 Pick the model with lowest p-value, if p-value less than SL_{in} then append that feature to dataset

step-3 Find feature corresponds to highest p-value

step-4 If p-value$>$ SL_{out} goto step-5 else break

step-5 Remove that feature from Dataset and goto step-2

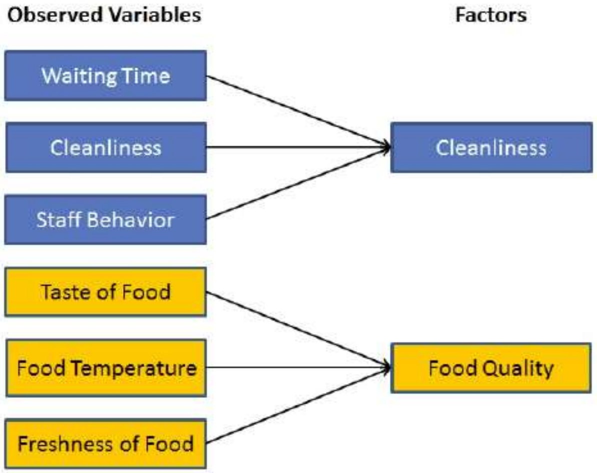
Step-6 Add remaining features one by one to above selected feature and run the model. goto step-3

Use SFS by enabling both forward and floating

- Suppose we have two variables: Income and Education. These variables will potentially have a high correlation as people with a higher education level tend to have significantly higher income, and vice versa.
- In the Factor Analysis technique, variables are grouped by their correlations, i.e., all variables in a particular group will have a high correlation among themselves, but a low correlation with variables of other group(s). Here, each group is known as a factor.
- These factors are small in number as compared to the original dimensions of the data.

- √ FA is an exploratory data analysis method used to extract factors or latent variables from the set of dataset features
- √ It extract maximum common variance from the features and put them a common score
- √ Factorial Analysis is a linear statistical model. It is used to represent set of observed variables into set of unobserved latent factors

$$X_i = \beta_{i0} + \beta_{i1} F_1 + \beta_{i2} F_2 + e_i \tag{1}$$


Observed Variables
Factors
Waiting Time
Cleanliness
Staff Behavior
Cleanliness
Taste of Food
Food Temperature
Freshness of Food
Food Quality

Assumptions

- √ There are no outliers in the data
- √ Sample size should be greater than factors
- √ There should not be perfect multicollinearity
- √ There should not be homoscedasticity between the variables

Types of factor analysis

- $\surd$ Exploratory factor analysis
  - Any observed variable directly related with any factors
- $\surd$ Confirmatory factor analysis
  - Set of observed variable directly related with only a set of factors

The primary objective is convert more number of observed variables into less number of unobserved variables. This conversion is achieved in two steps

- $\surd$ Factor extraction
  - Factors are extracted using variance partitioning methods like PCA and CFA
- $\surd$ Factor Rotation
  - Rotation is used to convert factors into un-correlated factors (Varimax, Quartimax and Promax)

What is Factor

- A factor is a latent variable which describes association among observed variables
- The maximum number of variables is equal to number of observed variables
- Every factor explains certain variance in every observed variable

Factor Loading

- It shows the relationship between each observed variable and latent factors.
- It shows correlation coefficient for observed variable and factor

Eigen Values:
Eigen values represents variance explained by each factor from total variance **Commonalities:**
It is sum of squared loadings of each variables

$$h_i^2 = \sum_{j=1}^{m} l_{ij}^2 \tag{2}$$