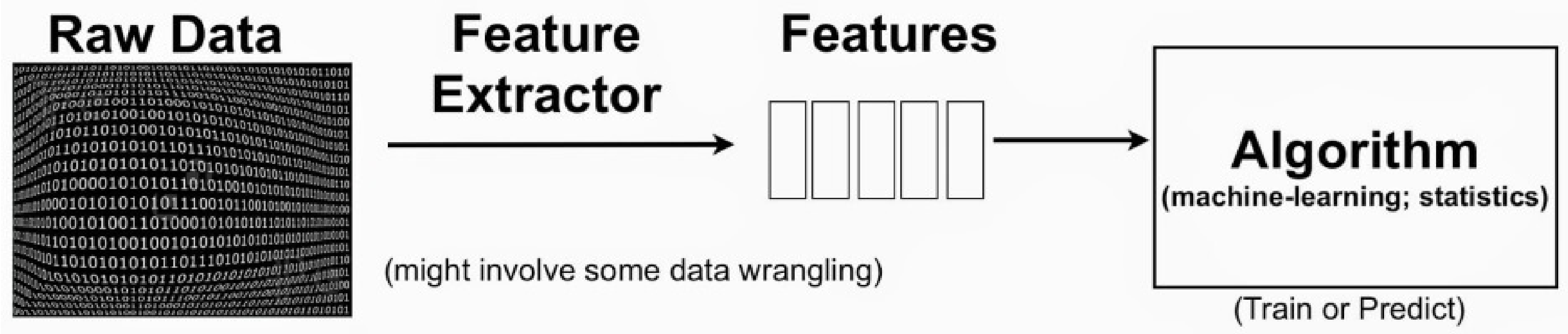
UNIT 3:
Data Engineering

Prepared by Nima Dema

GYALPOZHING
COLLEGE OF INFORMATION TECHNOLOGY

Feature Engineering

Feature Engineering?



Feature Engineering?

- **Feature engineering** is the act of extracting features from raw data and transforming them into formats that are suitable for the machine learning model.

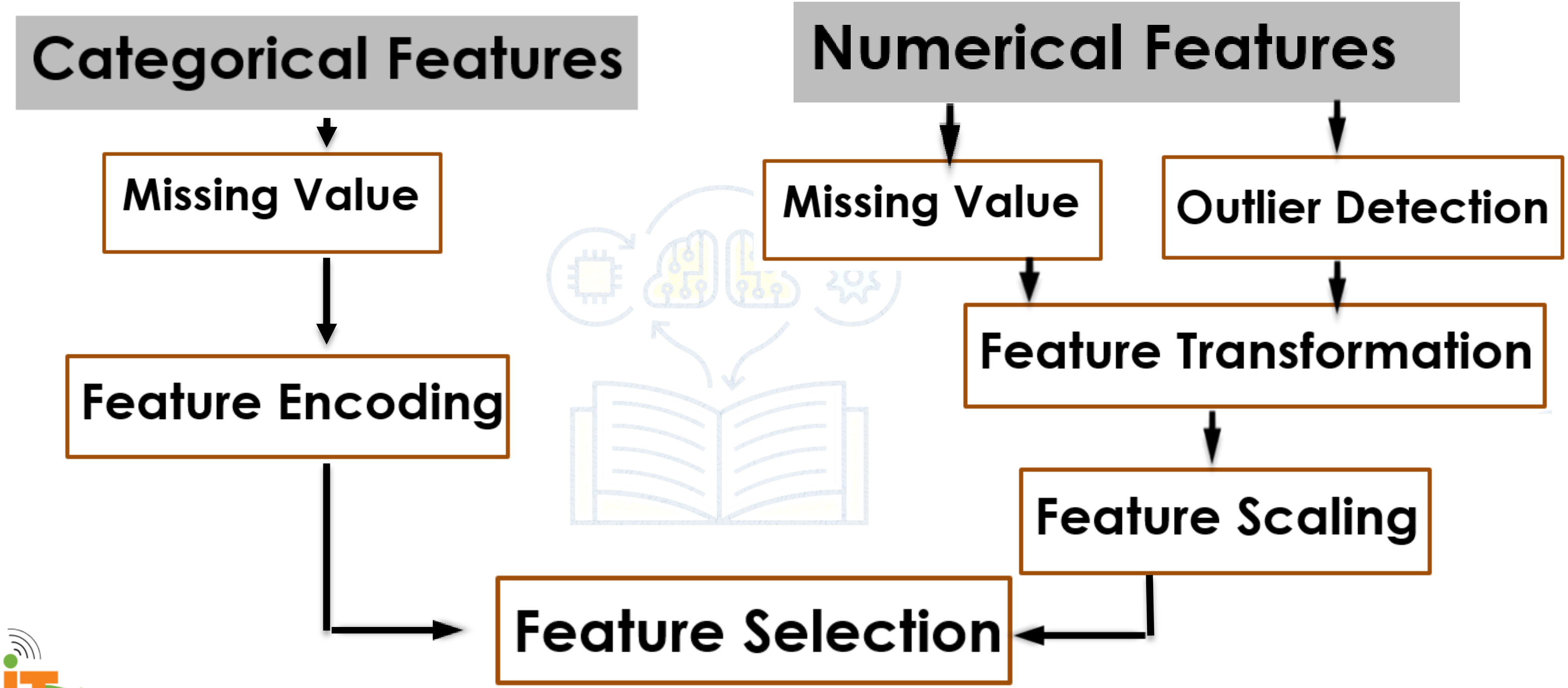
- An effective feature engineering implies;

  - Higher efficiency of the model

  - Easier algorithms that fits data

  - Easier for algorithms to detect pattern in the data

Feature Engineering?



Imputation

Categorical Features – Missing Value

	Id	Gender	Color
0	1	M	NaN
1	2	M	Red
2	3	F	Blue
3	4	NaN	Red
4	5	NaN	NaN
5	6	F	Red
6	7	M	Green
7	8	F	NaN

- Remove Missing Values
- Source Missing Values
- Imputation

Categorical Features – Missing Value: Imputation

```
categorical_cols = df.select_dtypes(include=['object','bool'])

from sklearn.impute import SimpleImputer

impute = SimpleImputer(strategy='most_frequent')

data = impute.fit_transform(categorical_cols)
```

- Use mode to impute categorical missing values.

OR

- Use pandas fillna() and choose bfill or ffill methods

Encoding Categorical Features

Categorical Features – Encoding

	Gender	Married
0	Male	No
1	Male	Yes
2	Male	Yes
3	Male	Yes
4	Male	No
5	Male	Yes

- The categories of a categorical variable are usually not numeric.
- Thus, an encoding method is needed to turn these nonnumeric categories into numbers.
- It is tempting to simply assign an integer, say from 1 to k, to each of k possible categories.
- But the resulting values would be orderable against each other, which should not be permissible for categories.

Categorical Features – One–Hot Encoding

- We use this categorical data encoding technique when the features are nominal (do not have any order).
- In one hot encoding, for each level of a categorical feature, we create a new variable. Each category is mapped with a binary variable containing either 0 or 1.
- Here, 0 represents the absence, and 1 represents the presence of that category.
- These newly created binary features are known as **Dummy variables.** The number of dummy variables depends on the levels present in the categorical variable.

Categorical Features – One–Hot Encoding

	Gender_Female	Gender_Male	Married_No	Married_Yes
0	0	1	1	0
1	0	1	0	1
2	0	1	0	1
3	0	1	0	1
4	0	1	1	0

GYALPOZHING
COLLEGE OF INFORMATION TECHNOLOGY

Categorical Features – One–Hot Encoding

```
from sklearn.preprocessing import OneHotEncoder

ohe = OneHotEncoder()

categorical = ohe.fit_transform(df[['Gender','Married']]).toarray()
categorical
```

```
y([[0., 1., 1., 0.],
   [0., 1., 0., 1.],
   [0., 1., 0., 1.],
   ...,
   [0., 1., 0., 1.],
   [0., 1., 0., 1.],
   [1., 0., 1., 0.]])
```

GYALPOZHING COLLEGE OF INFORMATION TECHNOLOGY

Categorical Features – One–Hot Encoding

- Use pandas get_dummies() method to create dummy variables.

```python
#creating dummy variables for nominal categorical variable using get_dummies()
categorical_columns=["Gender",'Married']
cdf = loandf[categorical_columns]
dummies_df = pd.get_dummies(cdf)
dummies_df.head()
```

	Gender_Female	Gender_Male	Married_No	Married_Yes
0	0	1	1	0
1	0	1	0	1
2	0	1	0	1
3	0	1	0	1
4	0	1	1	0

Categorical Features – Ordinal Encoding

Department	salary
sales	low
sales	medium
sales	medium
sales	low
sales	low

- We use this categorical data encoding technique when the categorical feature is ordinal.
- In this case, retaining the order is important. Hence encoding should reflect the sequence.
- In Ordinal encoding, each label is converted into an integer value.

Categorical Features – Ordinal Encoding

promotion_last_5years	Department	salary
0	sales	low
0	sales	medium
0	sales	medium
0	sales	low
0	sales	low

promotion_last_5years	Department	salary
0	sales	1.0
0	sales	2.0
0	sales	2.0
0	sales	1.0
0	sales	1.0

Categorical Features – Ordinal Encoding

```
from sklearn.preprocessing import OrdinalEncoder
oe =  OrdinalEncoder()
ndf['salary'] = oe.fit_transform(ndf[['salary']])
ndf.head()
```

- Ordinal Encoder's `fit_transform()` method expect 2D array as parameters.

Categorical Target – Label Encoding

Property_Area	Loan_Status
Urban	Y
Rural	N
Urban	Y
Urban	Y
Urban	Y
Urban	Y

- Classification problem: Target variable is categorical variables.

- Sklearn LabelEncoder is used to encode **categorical target** variable for classification problems.
- You may use LabelEncoder to encode categorical features as well.

Categorical Target – Label Encoding

Credit_History	Property_Area	Loan_Status
1.0	Urban	Y
1.0	Rural	N
1.0	Urban	Y
1.0	Urban	Y
1.0	Urban	Y

Credit_History	Property_Area	Loan_Status
1.0	Urban	1
1.0	Rural	0
1.0	Urban	1
1.0	Urban	1
1.0	Urban	1

Categorical Target – Label Encoding

```
from sklearn.preprocessing import LabelEncoder
le =  LabelEncoder()
loandf['Loan_Status'] = le.fit_transform(loandf.Loan_Status)
loandf.head()
```

- LabelEncoder's `fit_transform()` method expect 1D array as parameters.

Question Time

Question 1. What is feature engineering?

Question 2. Why do we need feature engineering?

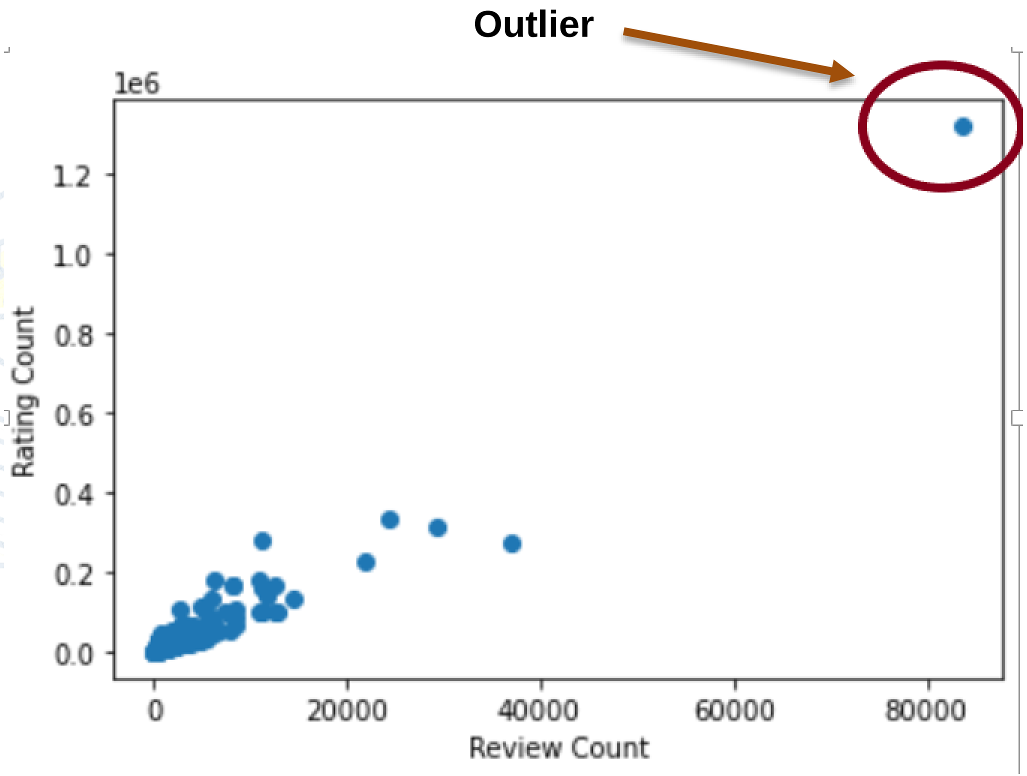
Question 3. What are the numerical feature engineering techniques?

Question 4. How do we handle missing value of numerical features?

Outlier Detection

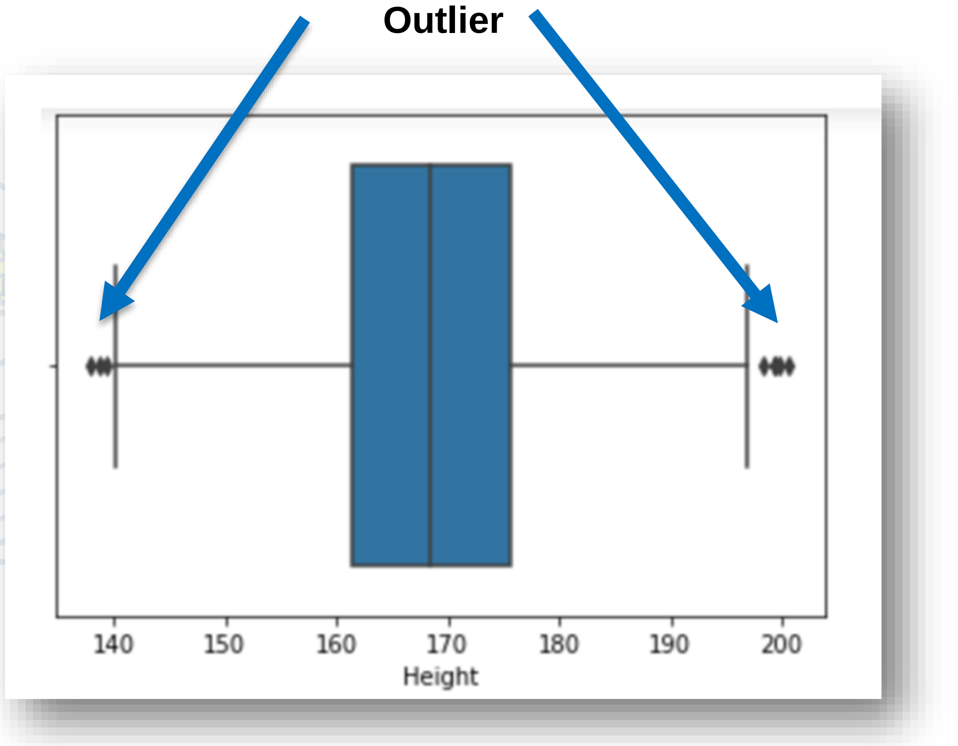
Numerical/Quantitative Features – Outlier Detection

- Outliers are observations in a dataset that don't fit in some way.
- Perhaps the most common or familiar type of outlier is the observations that are far from the rest of the observations.
- There are several ways to detect outlier and remove it.



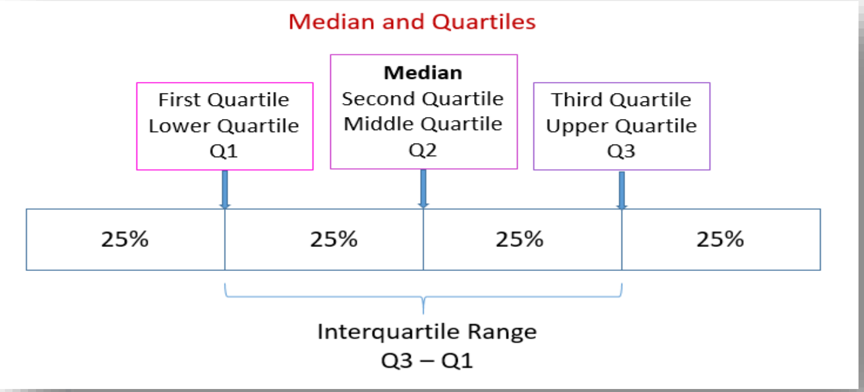
Numerical/Quantitative Features – Outlier Detection

- Outliers are observations in a dataset that don't fit in some way.
- Perhaps the most common or familiar type of outlier is the observations that are far from the rest of the observations.
- There are several ways to detect outlier and remove it.



Outlier Detection – Inter Quartile Range (IQR)

- The interquartile range rule is useful in detecting the presence of outliers.
- First step is to calculate interquartile range using IQR = Q3 - Q1.
- Once IQR is calculated, you need to determine upper and lower limit, typically the upper and lower whiskers of a box plot.
- All the values which falls below lower whiskers and above upper whiskers are considered as outliers.



$$\text{Upper limit} = Q3 + (IQR * 1.5)$$

$$\text{Lower limit} = Q1 - (IQR * 1.5)$$

GYALPOZHING COLLEGE OF INFORMATION TECHNOLOGY

Outlier Detection – Inter Quartile Range (IQR)

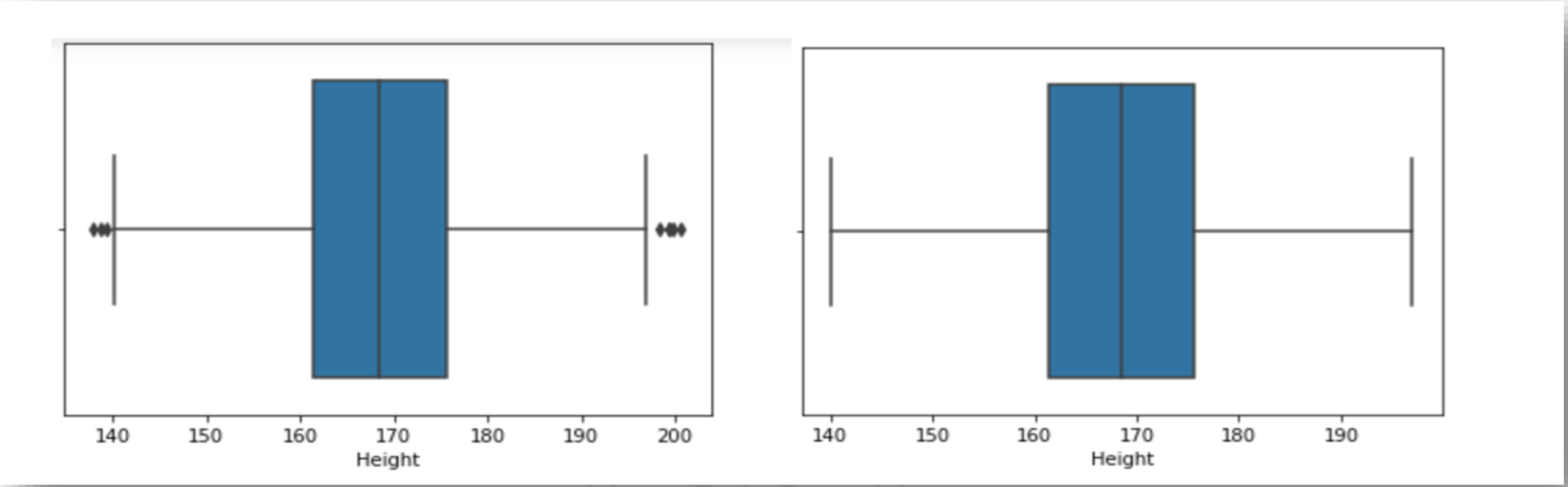
```
## calculating IQR and upper limit and lower limit to find outliers
Q1 = df.Height.quantile(0.25)
Q3 = df.Height.quantile(0.75)
IQR = Q3 - Q1
upperlimit = Q3 + (IQR * 1.5)
lowerlimit = Q1 - (IQR * 1.5)
```

```
## Drop all the rows containing height beyond lower and upper limit.
df1 = df[(df.Height > lowerlimit) & (df.Height < upperlimit)]
df1.shape
```

- Drop all the records/data that falls bellow lower limit and above upper limit.

GYALPOZHING COLLEGE OF INFORMATION TECHNOLOGY

Outlier Detection – Inter Quartile Range (IQR)

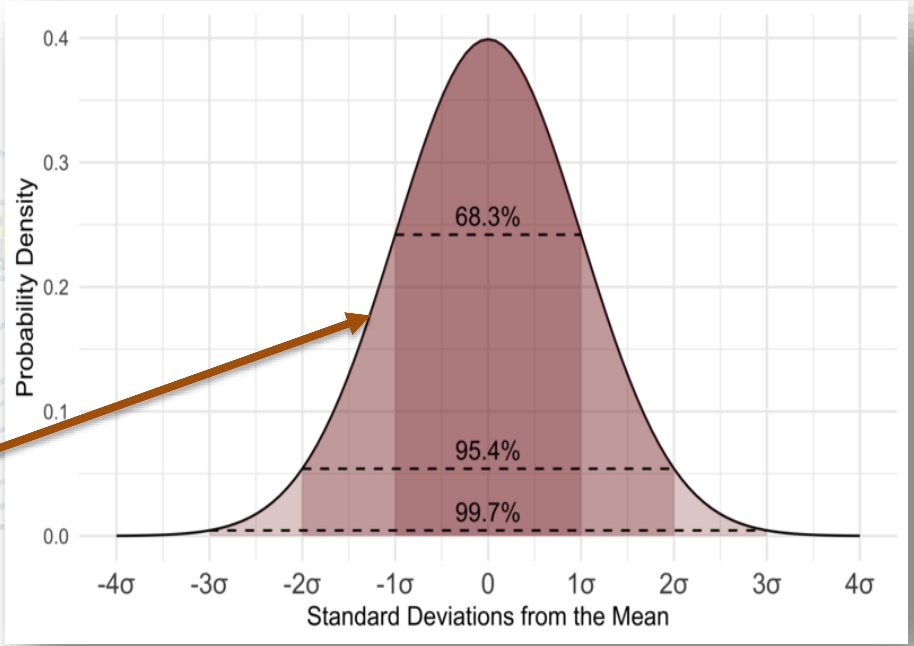


BEFORE

AFTER

Outlier Detection – Standard Deviation

- With Gaussian or Gaussian-like distribution, use the standard deviation of the sample as a cut-off for identifying outliers.
- Standard deviation from the mean can be used to reliably summarize the percentage of values in the sample.
- Within one standard deviation of the mean, covers 68% of the data, 2 standard deviation covers 95% of the data and 3 standard deviation covers 99.7% of the data.
- 3 SD is the Common cut-off



Outlier Detection – Standard Deviation

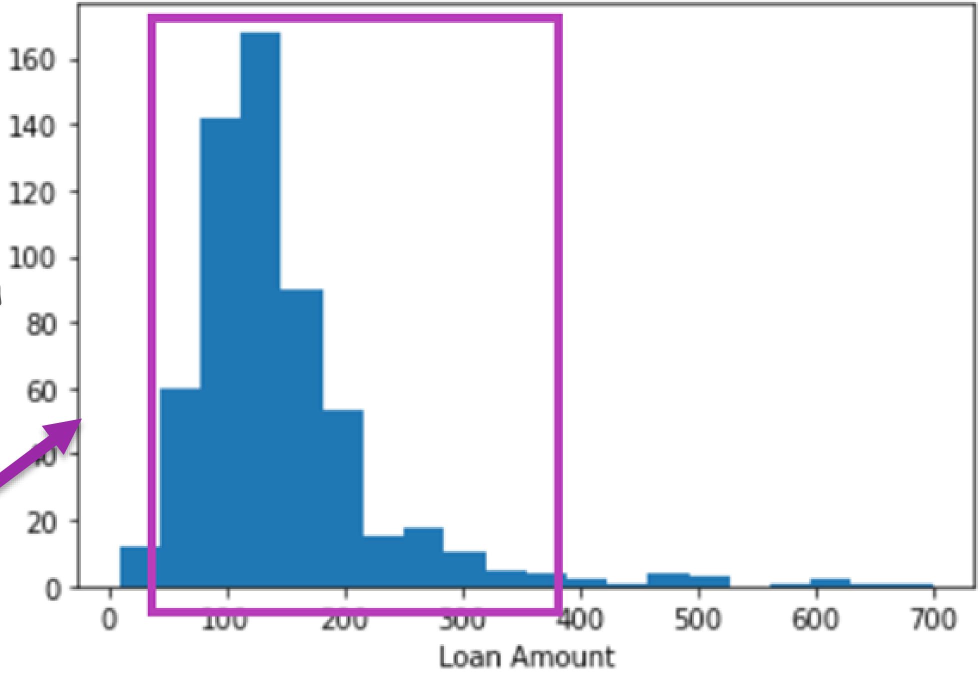
```python
# drop outliers which falls beyond 3 standard deviation range
std = df.Height.std()
mean = df.Height.mean()
upperlimit = mean + (3 * std)
lowerlimit = mean - (3 * std)
df2 = df[(df.Height>lowerlimit) & (df.Height<upperlimit)]
```

- Find standard Deviation and mean of the feature using std() and mean() methods of pandas series respectively.
- Set upper limit and lower limit taking 3 standard deviation as the cut off.
- Remove data which falls below lower limit and above upper limits from the dataset.

Feature Transformation

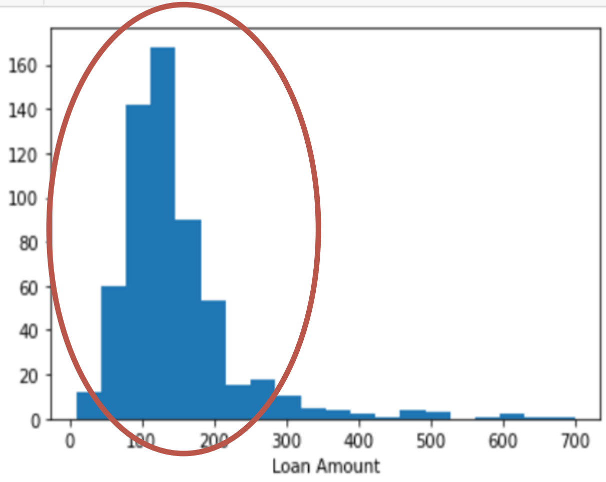
Numerical Features – Feature Transformation

- It's also important to consider the distribution of numeric features.
- The distribution of input features matters to some models more than others.
- Transformation is required to treat the skewed features and make them normally distributed.

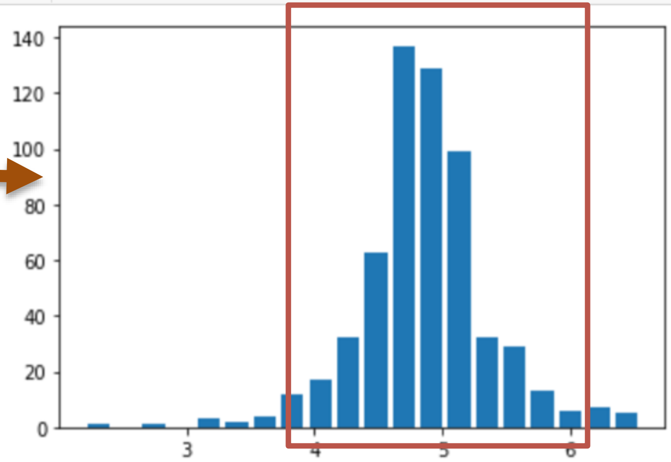


Feature Transformation – Logarithmic Transformation

```
plt.hist(loandf.LoanAmount,bins=20)
plt.xlabel("Loan Amount")
plt.show()
```




```
# Apply log transformation
x = np.log(loandf['LoanAmount'])
plt.hist(x,bins=20,rwidth=0.8)
plt.show()
```

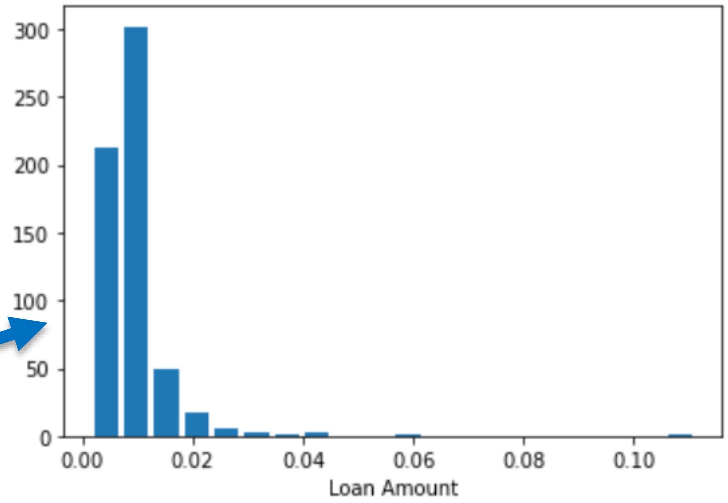


- Use numpy log() method to apply logarithmic Transformation.

Feature Transformation – Reciprocal Transformation

- Use **np.reciprocal()** method to use reciprocal transformation.
- The choice of the transformation technique depends on the distribution of the data.
- For **loanAmount** feature, reciprocal transformation is not a good choice.

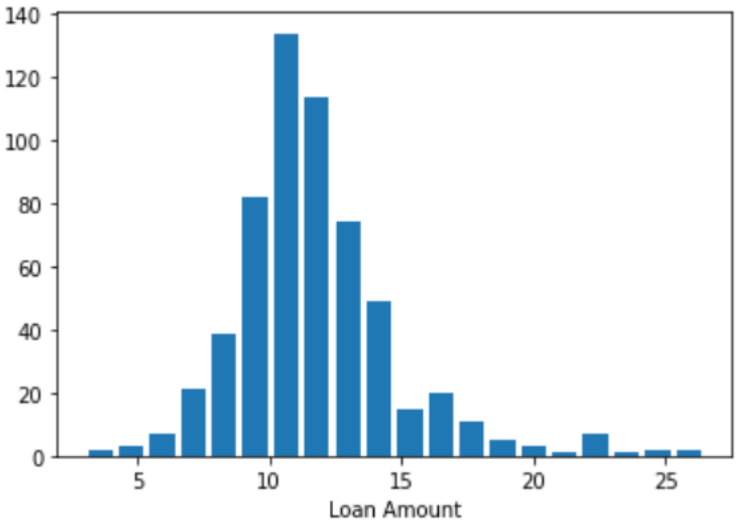
```python
# x = 1/loandf.LoanAmount
x = np.reciprocal(loandf.LoanAmount)
plt.hist(x,bins=20,rwidth=0.8)
plt.xlabel("Loan Amount")
plt.show()
```



Feature Transformation – Square root Transformation

- Use **`np.sqrt()`** method to apply square root transformation.

```python
# x = loandf.LoanAmount ** (1/2)
x = np.sqrt(loandf.LoanAmount)
plt.hist(x,bins=20,rwidth=0.8)
plt.xlabel("Loan Amount")
plt.show()
```

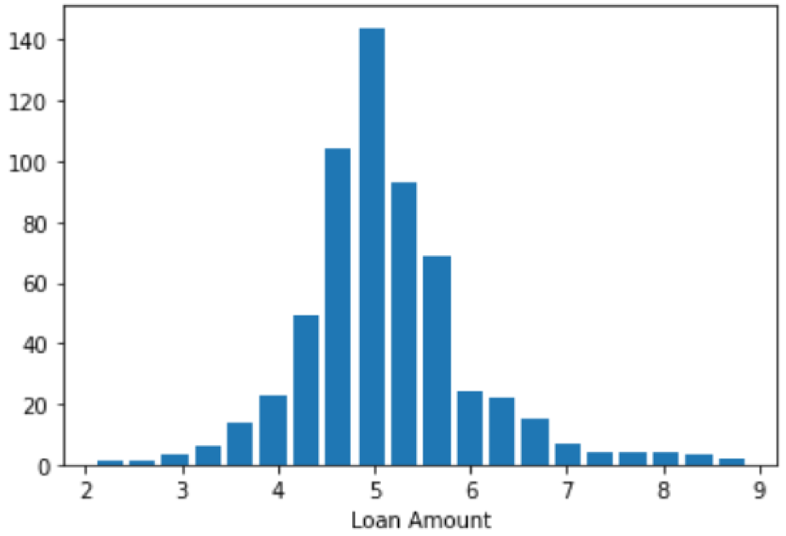


GYALPOZHING COLLEGE OF INFORMATION TECHNOLOGY

Feature Transformation – Cube root Transformation

- Use **np.cbrt()** method to apply Cube root transformation.

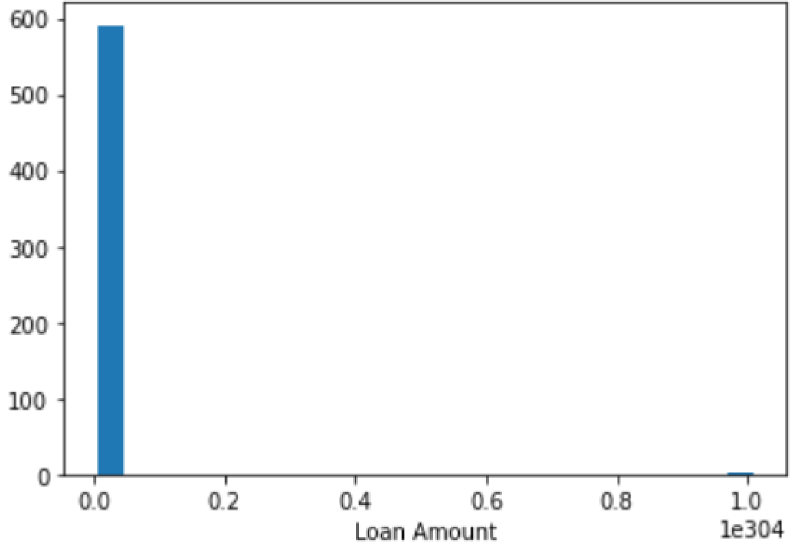
```python
# x = loandf.LoanAmount ** (1/3)
x = np.cbrt(loandf.LoanAmount)
plt.hist(x,bins=20,rwidth=0.8)
plt.xlabel("Loan Amount")
plt.show()
```



Feature Transformation – Exponential Transformation

- Use **np.exp()** method to apply exponential transformation.

```python
x = np.exp(loandf.LoanAmount)
plt.hist(x,bins=20,rwidth=0.8)
plt.xlabel("Loan Amount")
plt.show()
```



Feature Scaling

Numerical Features – Feature Scaling/Standardization

- Next, consider the scale of the features. What are the largest and the smallest values? Do they span several orders of magnitude?

	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	0.0	NaN	360.0	1.0
1	1508.0	128.0	360.0	1.0
2	0.0	66.0	360.0	1.0
3	2358.0	120.0	360.0	1.0
4	0.0	141.0	360.0	1.0

GYALPOZHING
COLLEGE OF INFORMATION TECHNOLOGY

Numerical Features – Feature Scaling/Standardization

- Machine learning algorithm just sees number.
- If there is a vast difference in the range say few ranging in thousands and few ranging in the tens, and it makes the underlying assumption that higher ranging numbers have superiority of some sort.
- So these more significant number starts playing a more decisive role while training the model.
- few algorithms like neural network gradient descent converge much faster with feature scaling than without it

Linear & Logistic Regression, KMeans/ KNN, Neural Networks, *PCA* will benefit from scaling.

Tree-Based Algorithms, Decision Tree, Random Forest, Boosted Trees(GBM, light GBM, xgboost) may not benefit from scaling.

Feature Scaling – Min Max Scaler

- Transform features by scaling each feature to a given range.
- This estimator scales and translates each feature individually such that it is in the given range on the training set.
- This Scaler shrinks the data within the range of -1 to 1 if there are negative values.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

GYALPOZHING COLLEGE OF INFORMATION TECHNOLOGY

Feature Scaling – Min Max Scaler

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
numeric = scaler.fit_transform(numericdf)
```

	ApplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	0.070489	0.172214	0.74359	1.0
1	0.054830	0.172214	0.74359	1.0
2	0.035250	0.082489	0.74359	1.0
3	0.030093	0.160637	0.74359	1.0
4	0.072356	0.191027	0.74359	1.0

Scaled Features

Feature Scaling – Standard Scaler

- The Standard Scaler assumes data is normally distributed within each feature and scales them such that the distribution centered around 0, with a standard deviation of 1.

- If data is not normally distributed, this is not the best Scaler to use.

$$x_{new} = \frac{x - \mu}{\sigma}$$

Feature Scaling – Standard Scaler

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
numeric = scaler.fit_transform(numericdf)
```

```
standardf = pd.DataFrame(numeric,columns=numeric_columns)
standardf.head()
```

	ApplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	0.072991	-0.217057	0.273231	0.433152
1	-0.134412	-0.217057	0.273231	0.433152
2	-0.393747	-0.947774	0.273231	0.433152
3	-0.462062	-0.311343	0.273231	0.433152
4	0.097728	-0.063843	0.273231	0.433152

Scaled Features

GYALPOZHING
COLLEGE OF INFORMATION TECHNOLOGY

Feature Scaling – Robust Scaler

- This Scaler is **robust** to outliers.
- If our data contains many **outliers**, scaling using the mean and standard deviation of the data won't work well.
- This Scaler removes the median and scales the data according to the quantile range (defaults to IQR: Interquartile Range).

Feature Scaling – Robust Scaler

```
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
numeric = scaler.fit_transform(numericdf)
```

```
robustdf = pd.DataFrame(numeric,columns=numeric_columns)
robustdf.head()
```

	ApplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	0.698029	0.000000	0.0	0.0
1	0.264096	0.000000	0.0	0.0
2	-0.278492	-0.928839	0.0	0.0
3	-0.421422	-0.119850	0.0	0.0
4	0.749786	0.194757	0.0	0.0

Scaled Features

GYALPOZHING
COLLEGE OF INFORMATION TECHNOLOGY

Feature Selection

Feature Selection

- **Feature selection** is the process of reducing the number of input variables when developing a predictive model.
- It is desirable to reduce the number of input variables
  - reduce the **computational cost of modeling**
  - **overfitting the model**
  - in some cases, to **improve the performance** of the model.

Curse of Dimensionality

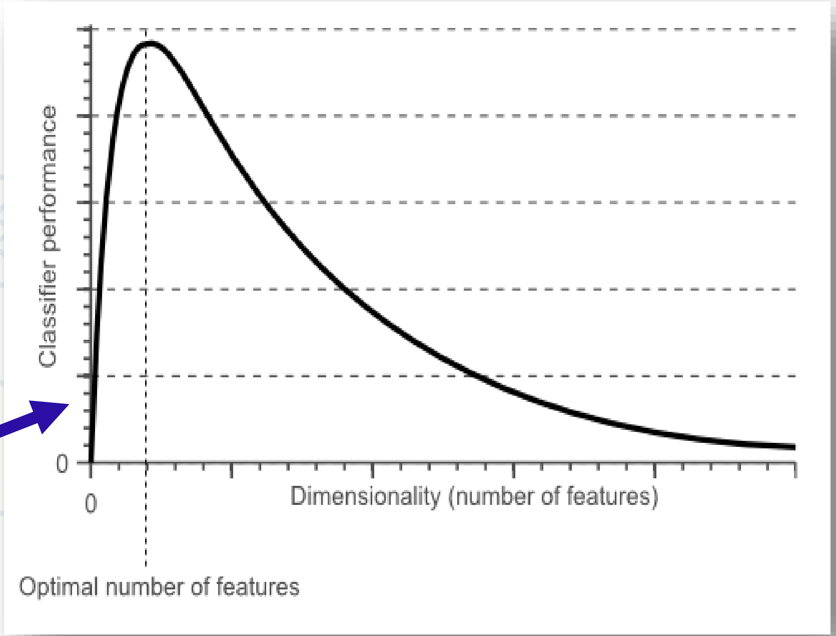
- **The curse of dimensionality** is a problem that arises when we are working with a lot of data having multiple features or we can say it as high dimensional data

Feature Selection

Curse of Dimensionality

- With high-dimensional, very challenging to identify meaningful patterns and it also degrades the machine learning model's accuracy while decreasing the computation speed as well.
- With the increase in dimensions, there are more chances for the occurrence of **multicollinearity** as well.

Hughes phenomenon shows that as the number of features increases, the classifier's/regressor performance increases as well until we reach the optimal number of features. Adding more features based on the same size as the training set will then degrade the model's performance.



GYALPOZHING COLLEGE OF INFORMATION TECHNOLOGY

Feature Selection Techniques

- There are several feature selection techniques you can used for choosing best features for your machine learning problems.

- The feature selection techniques that uses output/target feature to select the best features are called supervised selection models.

There are divided into:

→Filter method

→Wrapper Method

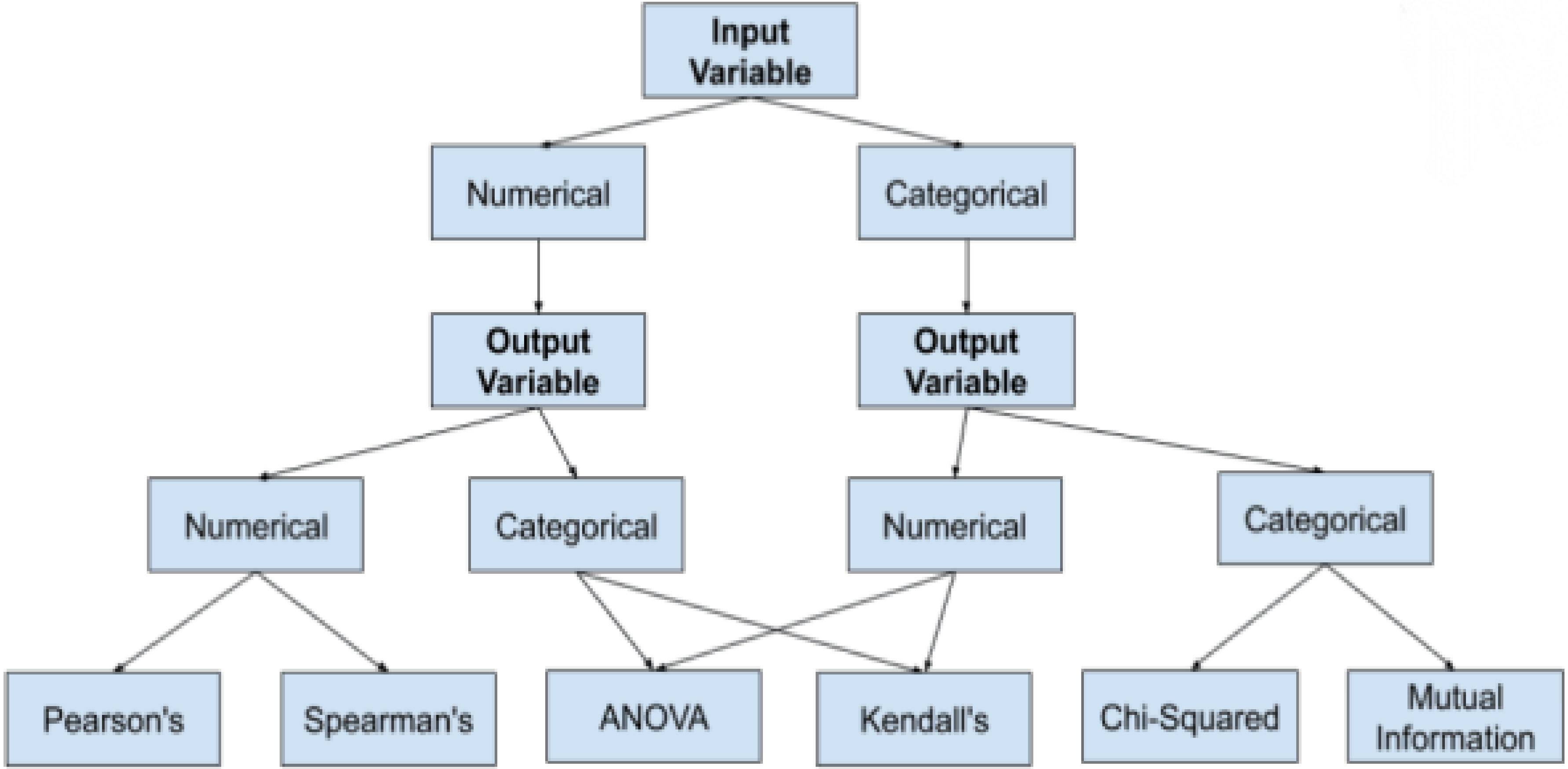
Feature Selection – Filter Method

- Features are dropped based on their relation to the output, or how they are correlating to the output.

- Use correlation to check if the features are positively or negatively correlated to the output feature and drop accordingly.

- Features are selected using some of the statistical test such as Pearson's correlation, Chi-Square Test etc.

- The choice of statistical test is determined by the type of input and output features.


Input Variable
Numerical
Categorical
Output Variable
Output Variable
Numerical
Categorical
Numerical
Categorical
Pearson's
Spearman's
ANOVA
Kendall's
Chi-Squared
Mutual Information

Feature Selection – Filter Method (Univariate Selection)

- Sklearn provides SelectKBest class to select k features most related to target feature.

SelectKBest(*score_func=<function f_classif>, *, k=10)*

- For regression: `f_regression`, `mutual_info_regression`
- For classification: `chi2`, `f_classif`, `mutual_info_classif`

Feature Selection – Filter Method (Univariate Selection)

Case 1: Numerical input and numerical output

- The most common techniques are to use a correlation coefficient, such as **Pearson's** for a linear correlation.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression

#define feature selection
fs = SelectKBest(score_func=f_regression,k=10)

#apply feature selection
X_selected = fs.fit_transform(bdf,boston.target)
```

Feature Selection – Filter Method (Univariate Selection)

Case 2: Numerical input and categorical output or Categorical Input and Numerical Output

- This section demonstrates feature selection for a classification problem that as numerical inputs and categorical outputs. Feature selection is performed using **ANOVA F** measure via the **f_classif()** function.

```
from sklearn.feature_selection import f_classif
fs = SelectKBest(score_func=f_classif,k=3)
X_selected = fs.fit_transform(idf,iris.target)
X_selected.shape
```

GYALPOZHING
COLLEGE OF INFORMATION TECHNOLOGY

Feature Selection – Filter Method (Univariate Selection)

Case 3: Categorical input and categorical output

- Pearson's chi-squared statistical hypothesis test is an example of a test for independence between categorical variables.

- This scikit-learn machine learning provides an implementation of the chi-squared test in the chi2() function..

```
from sklearn.feature_selection import chi2
fs = SelectKBest(score_func=chi2,k=8)
X_selected = fs.fit_transform(X,y)

```

Feature Selection – Wrapper Method

- Wrappers require some method to search the space of all possible subsets of features, assessing their quality by learning and evaluating the model with that feature subset.

- The feature selection process is based on a specific machine learning algorithm that we are trying to fit on a given dataset.

- It follows a greedy search approach by evaluating all the possible combinations of features against the evaluation criterion.

Wrapper Method – Forward Feature Selection

- Forward feature selection starts with the evaluation of each individual feature, and adds features which results in the best performance on selected algorithm of the model.

- All possible combinations of the that selected feature and a subsequent feature are evaluated, and a second feature is selected, and so on, until the required predefined number of features is selected.

- The best algorithm is chosen based on the evaluation metrics (scoring parameter given below).

```
class sklearn.feature_selection.SequentialFeatureSelector(estimator, *, n_features_to_select='warn', tol=None, direction='forward', scoring=None, cv=5, n_jobs=None)
```

Steps to perform Forward Feature Selection

https://www.analyticsvidhya.com/blog/2021/04/forward-feature-selection-and-its-implementation/

1. Train n model using each feature (n) individually and check the performance
2. Choose the variable which gives the best performance
3. Repeat the process and add one variable at a time
4. Variable producing the highest improvement is retained
5. Repeat the entire process until there is no significant improvement in the model's performance

GYALPOZHING
COLLEGE OF INFORMATION TECHNOLOGY

Wrapper Method – Forward Feature Selection

```
#import SequentialFeatureSelector

from sklearn.feature_selection import SequentialFeatureSelector

#create estimator or model
model = LinearRegression()

#create SequentialFeatureSelector object provide estimator, number of feature to
#select and feature selection technique to use

ffs = SequentialFeatureSelector(model,n_features_to_select=3,direction='forward')

#call fit_transform method to fit teh data and perform feature selection
ffs.fit_transform(bdf,boston.target)

#Check selected columns
ffs.get_feature_names_out(bdf.columns)
```

Wrapper Method – Backward Feature Elimination

- Starts with the entire set of features and works backward from there, removing features to find the optimal subset of a predefined size.

```
class sklearn.feature_selection.SequentialFeatureSelector(estimator, *, n_features_to_select='warn', tol=None, direction='backward', scoring=None, cv=5, n_jobs=None)
```

Wrapper Method – Backward Feature Elimination

```
#create SequentialFeatureSelector object provide estimator, number of feature to
#select and feature selection technique to use

bfs = SequentialFeatureSelector(model,n_features_to_select=3,direction='backward')

#call fit_transform method to fit teh data and perform feature selection
bfs.fit_transform(bdf,boston.target)

#Check selected columns
bfs.get_feature_names_out(bdf.columns)
```

Check the following blog:

1. Forward Selection:
   https://www.analyticsvidhya.com/blog/2021/04/forward-feature-selection-and-its-implementation/

2. Backward Selection:
   https://www.analyticsvidhya.com/blog/2021/04/backward-feature-elimination-and-its-implementation/?utm_source=blog&utm_medium=Forward_Feature_Elimination

GYALPOZHING COLLEGE OF INFORMATION TECHNOLOGY

THANK YOU ☺