Supervised Learning

Algorithm	Sklearn class	Parameters	Type	Evaluation Metrics
Linear Regression	LinearRegression	--	Regression	MAE, MSE, R2
Ridge	Ridge	alpha	Regression	MAE, MSE, R2
Lasso	Lasso	alpha	Regression	MAE, MSE, R2
Elastic Net	ElasticNet	alpha, l1_ratio	Regression	MAE, MSE, R2
Logistic Regression	LogisticRegression	penalty, C, l1 _ratio	Classification	Accuracy Score, Recall, Precision, F1-Score, ROC-AUC, Log Loss
K-NN Classifier	KNeighborsClassifier	n-neighbors	Classification	Accuracy Score, Recall, Precision, F1-Score, ROC-AUC, Log Loss
K-NN Regressor	KNeighborsRegressor	n-neighbors	Regression	MAE, MSE, R2
Naïve Bayes with categorical features	BernoulliNB		Classification	Accuracy Score, Recall, Precision, F1-Score, ROC-AUC, Log Loss
Naïve Bayes with numerical features	GaussianNB		Classification	Accuracy Score, Recall, Precision, F1-Score, ROC-AUC, Log Loss
Discriminant Analysis	LinearDiscriminant QuadraticDiscriminant		Classification	Accuracy Score, Recall, Precision, F1-Score, ROC-AUC, Log Loss
Support Vector Machines Classifier	SVC	kernel, C, gamma, degree, coef0	Classification	Accuracy Score, Recall, Precision, F1-Score, ROC-AUC, Log Loss
Decision Trees Classifier	DecisionTreeClassifier	max_depth, min_samples_ split, min_samples_ leaf	Classification	Accuracy Score, Recall, Precision, F1-Score, ROC-AUC, Log Loss
Decision Trees Regressor	DecisionTreeRegressor	max_depth, min_samples_ split, min_samples_ leaf	Regression	MAE, MSE, R2
Bagging Classifier	BaggingClassifier	estimator, n_estimators	Classification	Accuracy Score, Recall, Precision, F1-Score, ROC-AUC, Log Loss
Bagging Regressor	BaggingRegressor	estimator, n_estimators	Regression	MAE, MSE, R2
Random Forest	RandomForestClassifier	max_features	Classification	Accuracy Score, Recall, Precision, F1-Score, ROC-AUC, Log Loss

Classifier				
Random Forest Regressor	RandomForestRegressor	max_features	Regression	MAE, MSE, R2
Boosting Classifiers	sklearn: GradientBoostingClassifier XGB: XGBClassifier lightGBM: LGBMClassifier Cat Boost: CatBoostClassifier	learning_rate, n_trees, max_depth	Classification	Accuracy Score, Recall, Precision, F1-Score, ROC-AUC, Log Loss
Boosting Regressors	sklearn: GradientBoostingRegressor XGB: XGBRegressor lightGBM: LGBMRegressor Cat Boost: CatBoostRegressor	learning_rate, n_trees, max_depth	Regression	MAE, MSE, R2
Stacking Classifier	StackingClassifier	estimators, passthrough	Classification	Accuracy Score, Recall, Precision, F1-Score, ROC-AUC, Log Loss
Stacking Regressor	StackingRegressor	estimators, passthrough	Regression	MAE, MSE, R2
Time Series			Regression	MAE, MSE, R2

Unsupervised Learning

Algorithm	Function	Purpose	Finding/Evaluation
Agglomerative Clustering	AgglomerativeClustering	Cluster the data	Silhouette score
K-means clustering	Kmeans	Cluster the data	Silhouette score
Principal Component Analysis	PCA	Feature extraction and reduction	Variation explained
Association Rules Mining	Mlxtend.associationrules	Rules Extraction	Confidence, Lift Ratio
Recommender Systems	Surprise.KNNBasic & surprise.prediction_algorithms.matrix_factorization.SVD	Recommendation engine	Recommendations

Column Transformation

03 May 2024 14:27

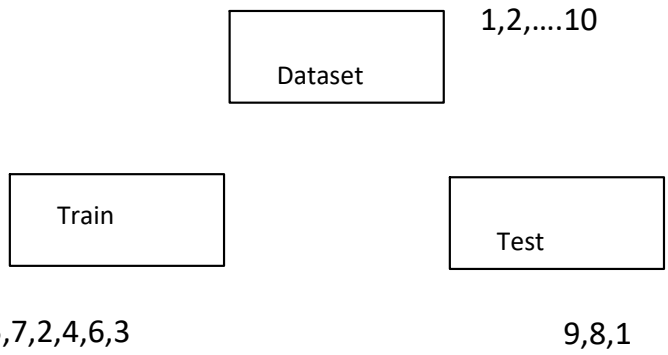
```
X = [['Male', 1],
     ['Female', 3],
     ['Female', 2]]
```

```
[[0., 1., 1., 0., 0.],
 [1., 0., 0., 0., 1.],
 [1., 0., 0., 1., 0.]]
```

make_column_transformer((ohc, make_column_selector(dtype_include=object)), ("passthrough", make_column_selector(dtype_exclude=object)) ,
verbose_feature_names_out=False)

K-Fold

26 April 2024 19:13



Hot Encoding / Dummying

```
pd.get_dummies(tel)
```

	TT_gt_100	Gender	Response
0	y	male	not bought
1	n	male	not bought
2	n	female	not bought
3	n	female	not bought
4	n	male	not bought
5	n	male	not bought
6	y	male	bought
7	y	female	bought
8	n	female	bought
9	y	female	bought

	TT_gt_100_n	TT_gt_100_y	Gender_female	Gender_male	Response_bought	Response_not bought
0	0	1	0	1	0	1
1	1	0	0	1	0	1
2	1	0	1	0	0	1
3	1	0	1	0	0	1
4	1	0	0	1	0	1
5	1	0	0	1	0	1
6	0	1	0	1	1	0
7	0	1	1	0	1	0
8	1	0	1	0	1	0
9	0	1	1	0	1	0

params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	ank_test_scor
{'alpha': 0.001}	-115.376	-98.9601	-114.421	-112.251	-104.011	-109.004	6.42352	4
{'alpha': 33.333999999999996}	-115.378	-98.9587	-114.421	-112.252	-104.009	-109.004	6.42471	3
{'alpha': 66.667}	-115.381	-98.9573	-114.421	-112.252	-104.007	-109.004	6.42589	2
{'alpha': 100.0}	-115.383	-98.956	-114.421	-112.252	-104.006	-109.004	6.42705	1

params	plit0_test_scor	plit1_test_scor	plit2_test_scor	plit3_test_scor	plit4_test_scor	nean_test_scor	std_test_score	ank_test_scor
{'alpha': 0.001}	0.541977	0.62408	0.631304	0.598889	0.636782	0.606607	0.0348204	3
{'alpha': 33.333999999999996}	0.541968	0.624085	0.631304	0.598888	0.636787	0.606607	0.0348256	1
{'alpha': 66.667}	0.541958	0.624091	0.631304	0.598887	0.636793	0.606607	0.0348307	2
{'alpha': 100.0}	0.541948	0.624096	0.631304	0.598886	0.636798	0.606607	0.0348358	4

Logistic Regression

27 April 2024 11:57

Linear Regression

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m$$

Coefficients are calculated by method of least squares

Logistic Regression

$$y = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_m x_m)}}$$

Coefficients are calculated by method of maximum likelihood

```
In [130]: print(accuracy_score(y_test, y_pred))
     ...: print(roc_auc_score(y_test, y_pred_prob[:,1]))
     ...: print(log_loss(y_test, y_pred_prob))
0.8
0.79
0.5654129213670671
```

```
In [133]: print(accuracy_score(y_test, y_pred))
     ...: print(roc_auc_score(y_test, y_pred_prob[:,1]))
     ...: print(log_loss(y_test, y_pred_prob))
0.75
0.7975
0.5533347765065828
```

OVR: One Vs Rest of All

Regularized Regression

30 April 2024 08:45

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(y-\hat{y})^2 + r\lambda\sum_{j=1}^{n}|w_j| + \frac{1-r}{2}\lambda\sum_{j=1}^{n}w_j^2$$

Lasso Ridge

Model	R2
LR	0.7442304540793144
Ridge	0.7442304381635699
Lasso	0.7445319572206965
ElasticNet	0.7444779079832952

```
{'alpha': array([1.000000e-03, 2.500075e+01, 5.000050e+01,
7.500025e+01,
        1.000000e+02]),
 'l1_ratio': array([0.001, 0.334, 0.667, 1.   ])}
```

params	plit0_test_scor	split1_test_score	plit2_test_scor	split3_test_score	plit4_test_scor	mean_test_score	std_test_score	rank_test_score
{'alpha': 0.001, 'l1_ratio': 0.001}	0.776059	0.71119	0.761415	0.714281	0.75817	0.744223	0.0264247	6
{'alpha': 0.001, 'l1_ratio': 0.334}	0.776213	0.711323	0.761427	0.714186	0.758005	0.744231	0.0264344	4
{'alpha': 0.001, 'l1_ratio': 0.667}	0.776363	0.711452	0.761435	0.714085	0.757833	0.744233	0.0264449	3
{'alpha': 0.001, 'l1_ratio': 1.0}	0.776508	0.711575	0.761437	0.713978	0.757655	0.74423	0.0264564	5
{'alpha': 25.00075, 'l1_ratio': 0.001}	0.125131	0.0392062	0.139884	0.0787034	0.174638	0.111512	0.0474971	12
{'alpha': 25.00075, 'l1_ratio': 0.334}	0.130392	0.0428284	0.149638	0.0831491	0.180679	0.117337	0.0488802	10
{'alpha': 25.00075, 'l1_ratio': 0.667}	0.142346	0.0542953	0.168209	0.0956927	0.193268	0.130762	0.0500409	9
{'alpha': 25.00075, 'l1_ratio': 1.0}	0.775339	0.710923	0.760928	0.716539	0.758661	0.744478	0.0258097	1
{'alpha': 50.000499999999995, 'l1_ratio': 0.001}	0.115103	0.0360106	0.122616	0.0732713	0.16149	0.101698	0.0431631	16
{'alpha': 50.000499999999995, 'l1_ratio': 0.334}	0.121252	0.0374693	0.132788	0.0762036	0.169796	0.107502	0.0460285	14
{'alpha': 50.000499999999995, 'l1_ratio': 0.667}	0.13027	0.0427165	0.149428	0.0830545	0.180534	0.117201	0.0488556	11

[-1.08011358e-01 4.64204584e-02 2.05586264e-02 2.68673382e+00
-1.77666112e+01 3.80986521e+00 6.92224640e-04 -1.47556685e+00
3.06049479e-01 -1.23345939e-02 -9.52747232e-01 9.31168327e-03
-5.24758378e-01] 36.45948838508963

[-1.04595278e-01 4.74432243e-02 -8.80467889e-03 2.55239322e+00

-1.07770146e+01 3.85400020e+00 -5.41453810e-03 -1.37265353e+00
 2.90141589e-01 -1.29116463e-02 -8.76074394e-01 9.67327945e-03
-5.33343225e-01] 31.597669818274188

[-0.06343729 0.04916467 -0.   0.   -0.   0.9498107
 0.02090951 -0.66879   0.26420643 -0.01521159 -0.72296636 0.00824703
-0.76111454] 41.05693374499339

[-0.08037077 0.05323951 -0.0126571   0.   -0.   0.93393555
 0.0205792 -0.76204391 0.30156906 -0.01643916 -0.7480458   0.00833878
-0.75842612] 42.22956397215435

[-1.08011358e-01 4.64204584e-02 2.05586264e-02 2.68673382e+00
 -1.77666112e+01 3.80986521e+00 6.92224640e-04 -1.47556685e+00
3.06049479e-01 -1.23345939e-02 -9.52747232e-01 9.31168327e-03
 -5.24758378e-01] 36.45948838508963

[-0.10205324 0.04856625 -0.03287066 2.29723414 -4.94178208 3.83169239
 -0.00990859 -1.28696176 0.27949223 -0.01351404 -0.81643051 0.00995699
 -0.54511331] 28.04237427218255

Polynomial

30 April 2024 12:41

Degree	R2
1	0.7133431144123451
2	0.6865974604825266
3	-8442.44531626403

Model	R2
Ridge	0.802391612960911 4
Lasso	0.8304349679612066
ElasticNet	0.8304349679612066

Model	R2
Ridge	0.8618588990766975
Lasso	0.8526437598479252
ElasticNet	0.8529945524897767

K-NN

01 May 2024 12:45

Log loss
w/o scaling : 0.3704012180788165 4
Standard: 0.3170340836795815
MM: 1.7425616149199414

Grid Search
w/o scaling:
{'n_neighbors': 8}
-1.2503592106282053

Standard Scaling:
{'KNN__n_neighbors': 10}
-0.3545562027841026

Min Max Scaling:
{'KNN__n_neighbors': 9}
-0.3541342613432673

Discriminant

02 May 2024 08:16

6.4 Bivariate Normal Distribution

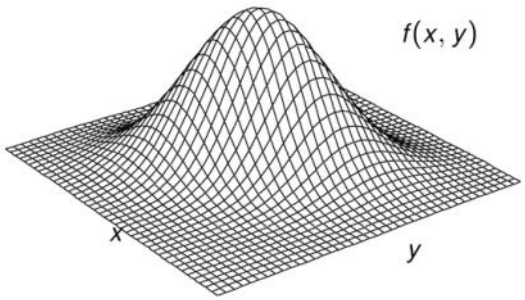
The continuous random variables X and Y with joint pdf

$$f(x,y)=\frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho^2}}\,e^{-\frac{1}{2(1-\rho^2)}\left[\left(\frac{x-\mu_X}{\sigma_X}\right)^2-2\rho\left(\frac{x-\mu_X}{\sigma_X}\right)\left(\frac{y-\mu_Y}{\sigma_Y}\right)+\left(\frac{y-\mu_Y}{\sigma_Y}\right)^2\right]}$$

on $\mathcal{A}=\{-\infty<x<\infty,\ -\infty<y<\infty\}$ with parameter space

$\Omega=\{(\mu_X,\mu_Y,\sigma_X,\sigma_Y,\rho)|-\infty<\mu_X<\infty,-\infty<\mu_Y<\infty,\sigma_X>0,\sigma_Y>0,-1<\rho<1\}$

are *bivariate normal random variables* with parameters μ_X, μ_Y, σ_X, σ_Y, and ρ



Screencast-O-Matic.com Joint Distributions Probability

Multivariate normal distribution

Multivariate normal distribution: Joint pdf

$$f(x_1,x_2,\ldots,x_n)=\frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}}\,e^{-\frac{1}{2}(x-\mu)'\boldsymbol{\Sigma}^{-1}(x-\mu)}\qquad(x_1,x_2,\ldots,x_n)\in\mathcal{A}$$

where

- random vector $X=(X_1,X_2,\ldots,X_n)'$
- vector of means $\boldsymbol{\mu}=(\mu_1,\mu_2,\ldots,\mu_n)'$
- variance–covariance matrix $\boldsymbol{\Sigma}$
- $x=(x_1,x_2,\ldots,x_n)'$
- $\boldsymbol{\Sigma}^{-1}$ is the inverse of the variance–covariance matrix
- $|\boldsymbol{\Sigma}|$ is the determinant of the variance–covariance matrix
- the support is

$$\mathcal{A}=\{(x_1,x_2,\ldots,x_n)\,|\,-\infty<x_i<\infty,\ \text{for } i=1,2,\ldots,n\}$$

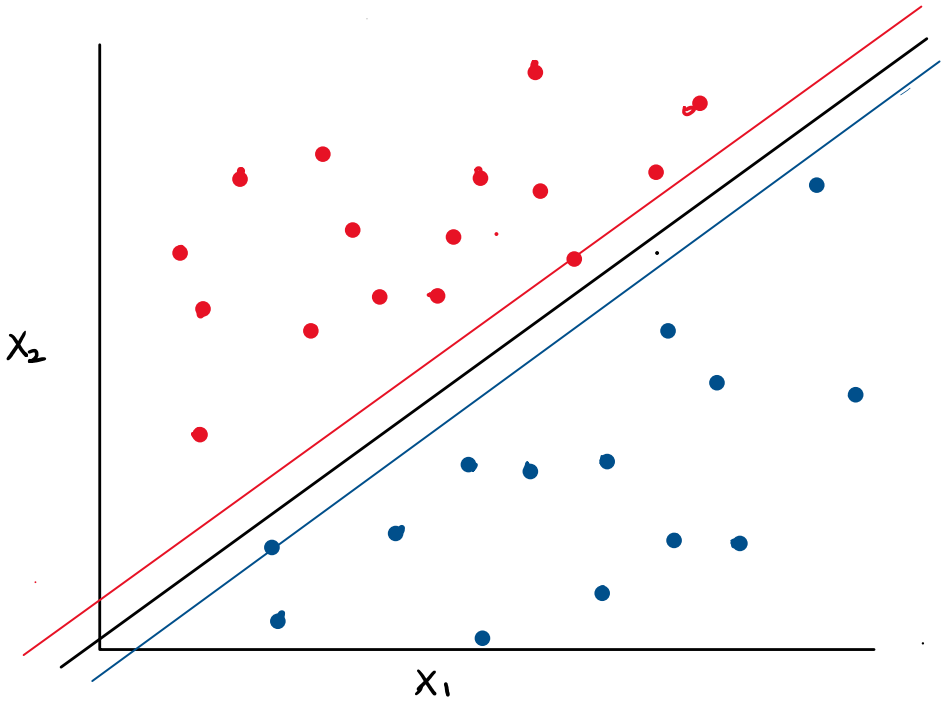
- the parameter space is

$$\Omega=\{(\boldsymbol{\mu},\boldsymbol{\Sigma})\,|\,\boldsymbol{\mu}\in\mathcal{R}^n,\ \boldsymbol{\Sigma}\text{ is an } n\times n \text{ positive-semidefinite matrix}\}$$

Screencast-O-Matic.com Joint Distributions Probability

$$\delta_i(x)=x\frac{\mu_i}{\sigma^2}-\frac{\mu_i^2}{2\sigma^2}+\log(\mathrm{P}(C_i))$$

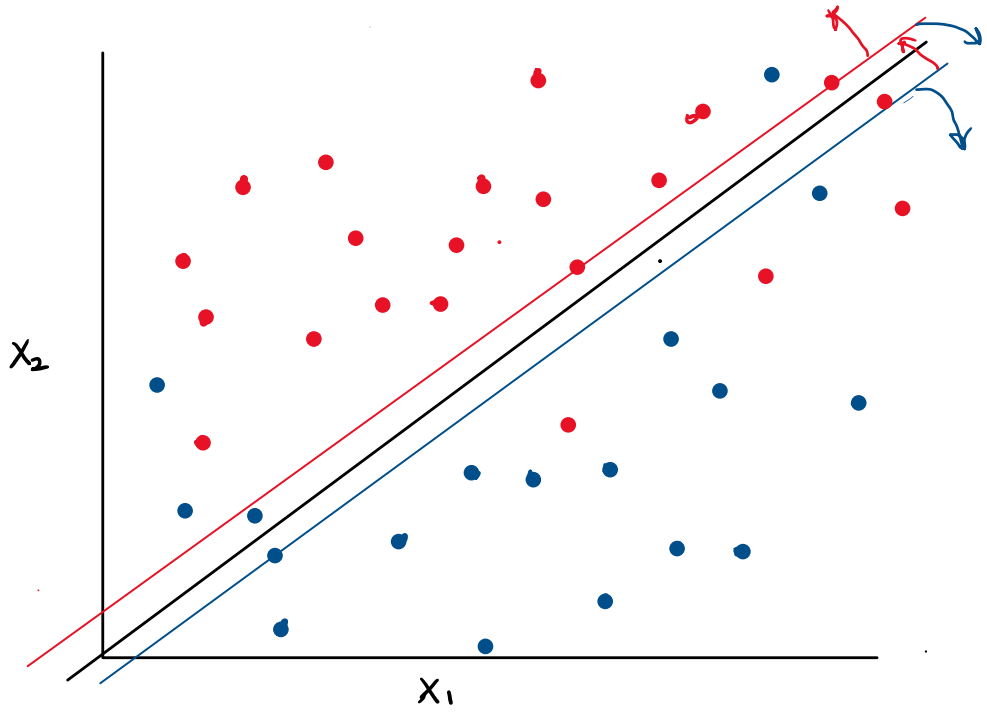
$$\delta_i(\bar{x}) = x^T \sum^{-1} \mu_i - \frac{1}{2}\mu_i^T \sum^{-1} \mu_i + \log(P(C_i))$$



Training Data

X_1	X_2	y
		1
		0
		0
		1
		0
		1
		0

Testing Data



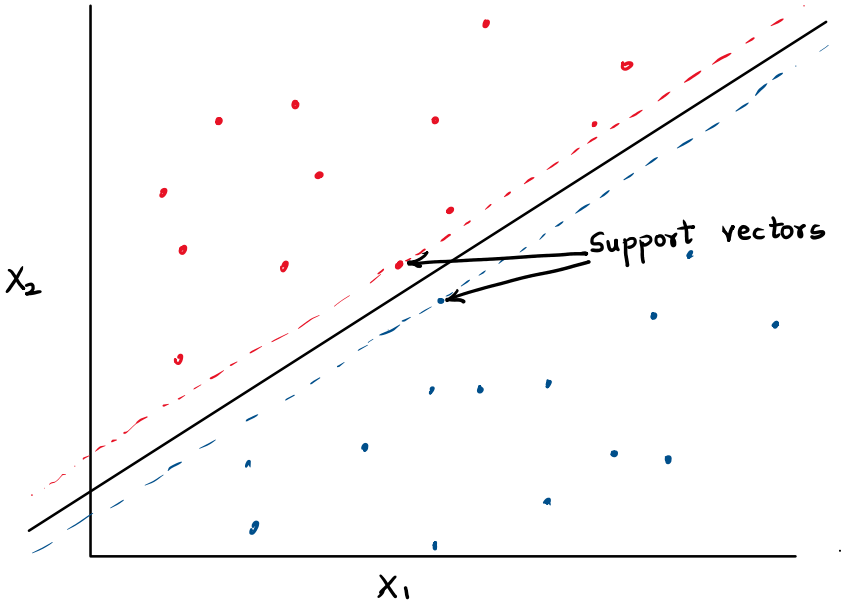
params	plit0_test_scor	plit1_test_scor	plit2_test_scor	plit3_test_scor	plit4_test_scor	nean_test_scor	std_test_score	ank_test_sc
{'C': 0.1}	0.705882	0.8125	0.875	0.875	0.6875	0.791176	0.0806615	5
{'C': 1}	0.705882	0.8125	0.875	0.875	0.75	0.803676	0.0673748	1
{'C': 0.5}	0.705882	0.8125	0.875	0.875	0.75	0.803676	0.0673748	1
{'C': 2}	0.705882	0.8125	0.875	0.875	0.75	0.803676	0.0673748	1
{'C': 3}	0.705882	0.8125	0.875	0.875	0.75	0.803676	0.0673748	1

SVM

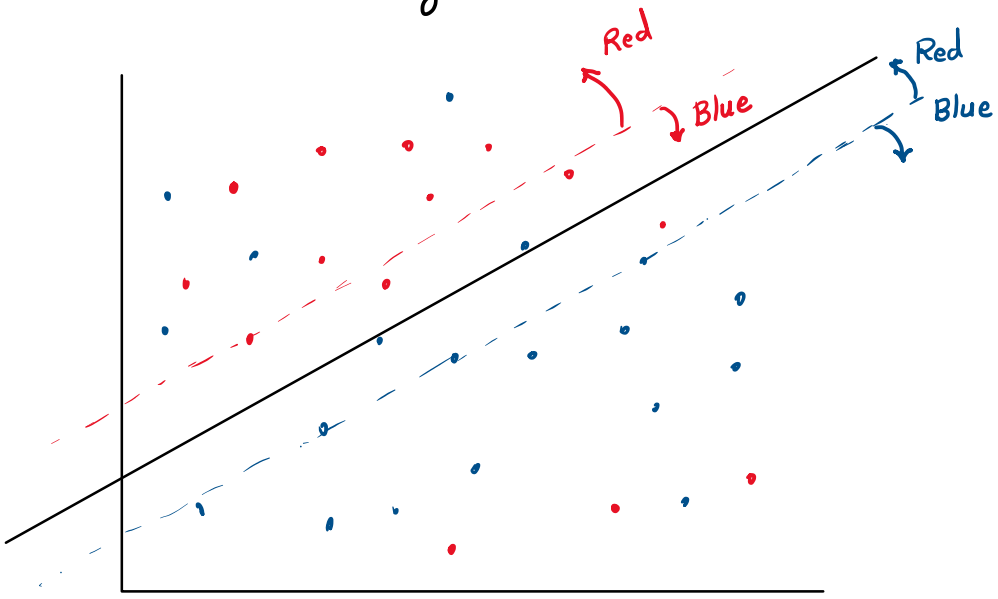
Sunday, March 10, 2024 10:40 AM

Maximum Margin Classifier

X_1	X_2	y
		1
		0
		0
		1
		0
		1
		0



$C \propto \frac{1}{\text{margin}}$

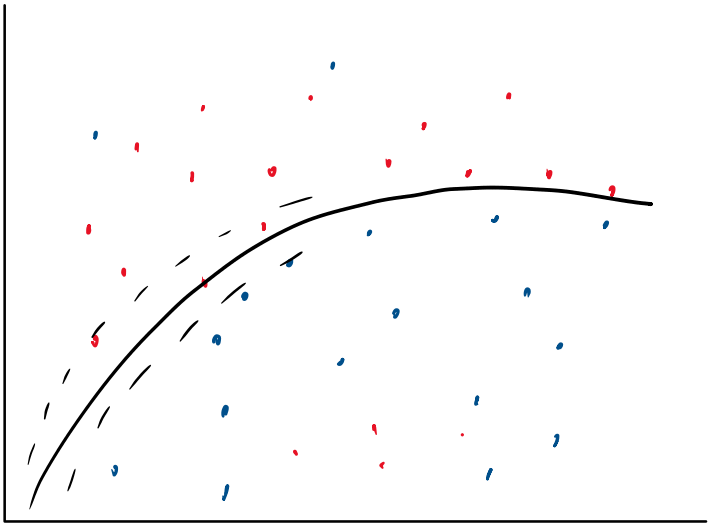


w/o scaling
{'C': 0.5357894736842106}
0.6075786282682835

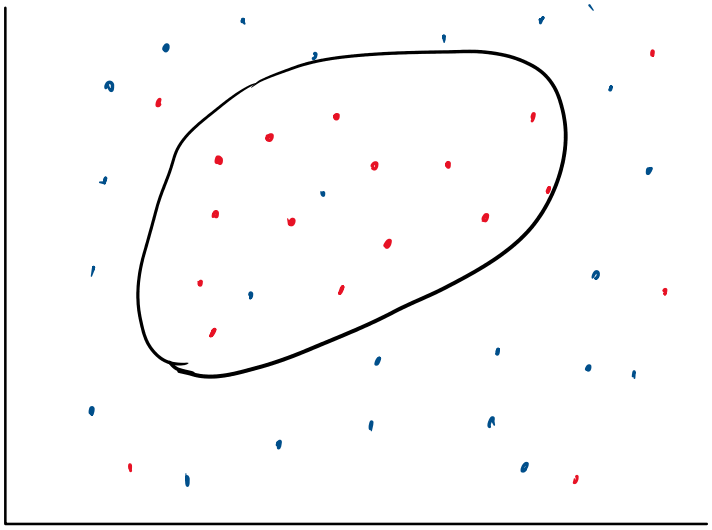
With scaling
{'SCL': StandardScaler(), 'SVM__C': 10.0}
0.6278684833407471

Polynomial : $a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_n x^n$

3^{rd} degree : $a_0 + a_1 x + a_2 x^2 + a_3 x^3$



$r \propto \frac{1}{size}$



Training

y	X_1 X_2 ... X_p	
A		40 obs
A		
⋮		
A		
B		30 obs
B		
⋮		
B		
C		50 obs
⋮		
C		

1) OVO : One Vs One classification

y	x_1 .... x_p
A	
B	70 obs

SVM_{AB}

y	x_1 .... x_p
A	
C	90 obs

SVM_{AC}

y	x_1 .... x_p
B	
C	80 obs

SVM_{BC}

m categories → ${}^{m}C_2$ SVMs

SVM_{AB} SVM_{AC} SVM_{BC}

Testing

	X_1 X_2 .... X_p	y_{AB}	y_{AC}	y_{BC}	$\hat{y}$
1)		A	A	B	A
2)		B	C	C	C
3)		A	C	B	A

y	X_1 X_2 ... X_p	
A		40 obs
A		
⋮		
A		
B		30 obs
B		
⋮		
B		
C		50 obs
⋮		
C		

2) OVR : One Vs Rest of All

y	X_1 .... X_p
A	
non A	: B or C

SVM_A

y	X_1 .... X_p
B	
non B	: A or C

SVM_B

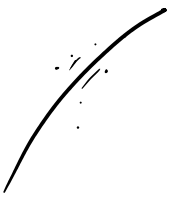
y	X_1 .... X_p
C	
non C	: B or A

SVM_C

Testing

X_1 X_2 ... X_p	$P(y=A)$	$P(y=B)$	$P(y=C)$	$\hat{y}$
	0.8	0.2	0.9	C

SVR :- epsilon

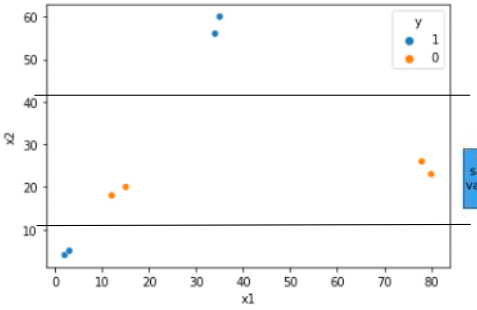


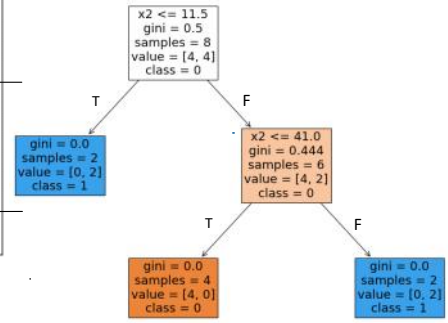
Linear Kernel Results:
Params: {'SCL': None, 'SVM__C': 0.26410526315789473}
Score: -0.46079499942791546
Polynomial Kernel Results:
Params: {'SCL': MinMaxScaler(), 'SVM__C': 0.26410526315789473, 'SVM__coef0': 3.0, 'SVM__degree': 2}
Score: -0.5097130472240468
Radial Kernel Results:
Params: {'SCL': MinMaxScaler(), 'SVM__C': 0.26410526315789473, 'SVM__gamma': 1.2507499999999998}
Score: -0.4809688740028147

Linear Kernel Results:
Params: {'SCL': None, 'SVM__C': 0.7903157894736842, 'SVM__decision_function_shape': 'ovo'}
Score: -0.9312604498988577
Polynomial Kernel Results:
Params: {'SCL': StandardScaler(), 'SVM__C': 0.26410526315789473, 'SVM__coef0': 3.0, 'SVM__decision_function_shape': 'ovo', 'SVM__degree': 3}
Score: -0.8912631607153948
Radial Kernel Results:
Params: {'SCL': None, 'SVM__C': 1.5796315789473683, 'SVM__decision_function_shape': 'ovo', 'SVM__gamma': 1.2507499999999998}
Score: -0.7937284699511382

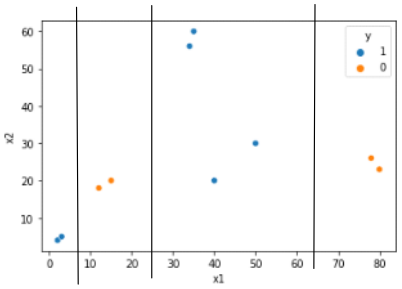
Trees

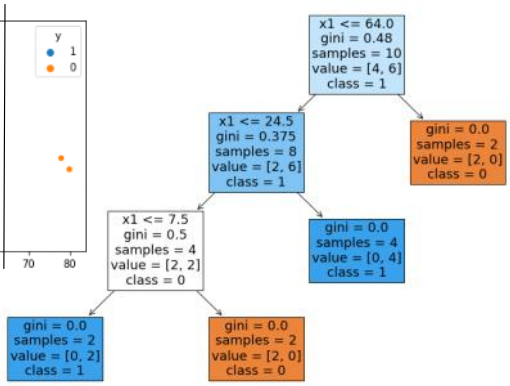
03 May 2024 08:11



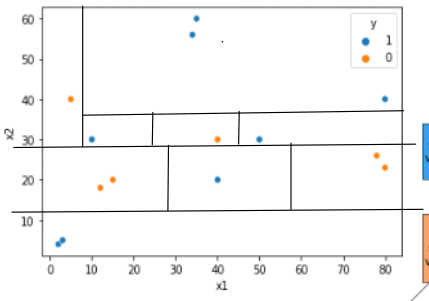


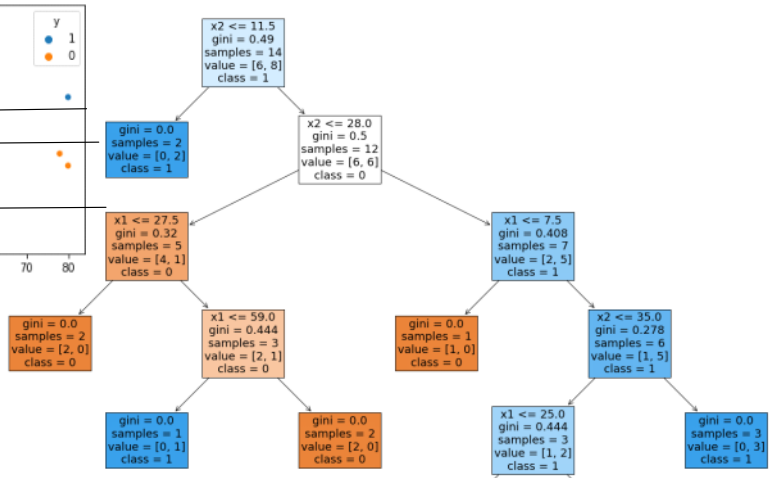
X1 = 40, X2 = 30 ==> 0
X1 = 30, X2 = 5 ==> 1



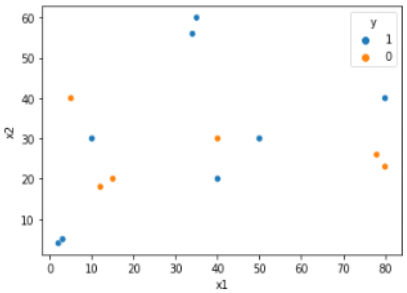


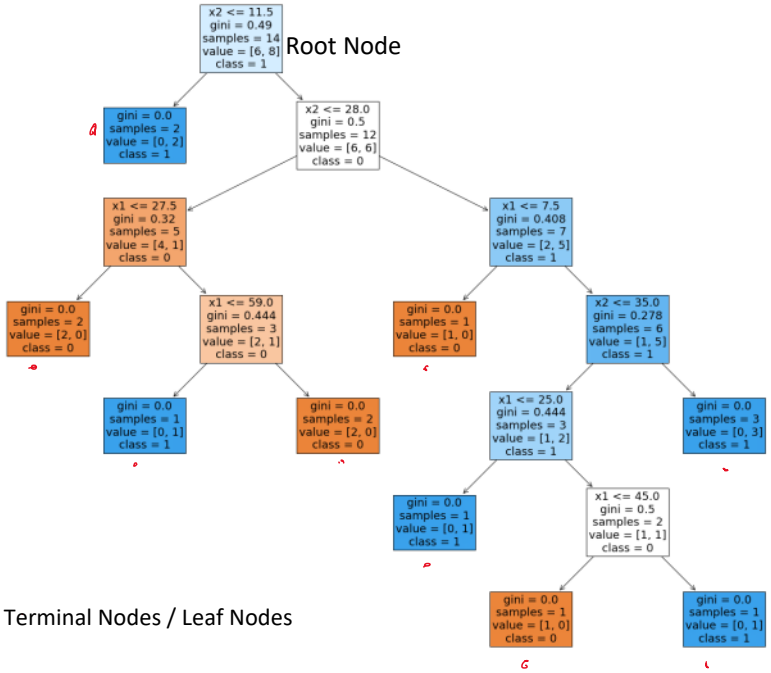
X1 = 40, X2 = 30 ==> 1
X1 = 30, X2 = 5 ==> 1

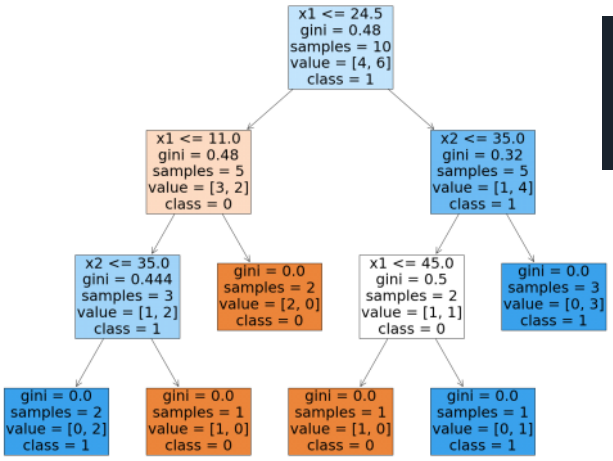




X1 = 40, X2 = 30 ==> 0
X1 = 30, X2 = 5 ==> 1

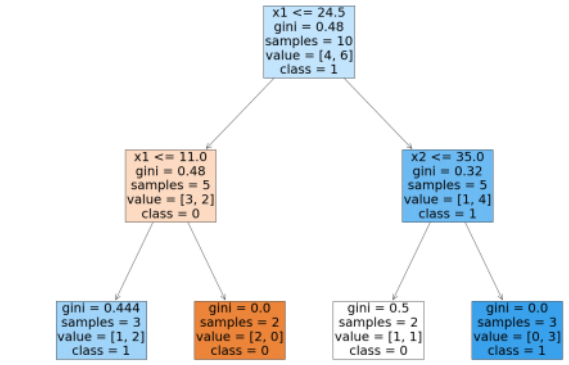






X_test

x1	x2	y	P(y=0)	P(y=1)
78	26	1	0	1
40	20	0	1	0
3	5	1	0	1
80	23	1	0	1



X_test

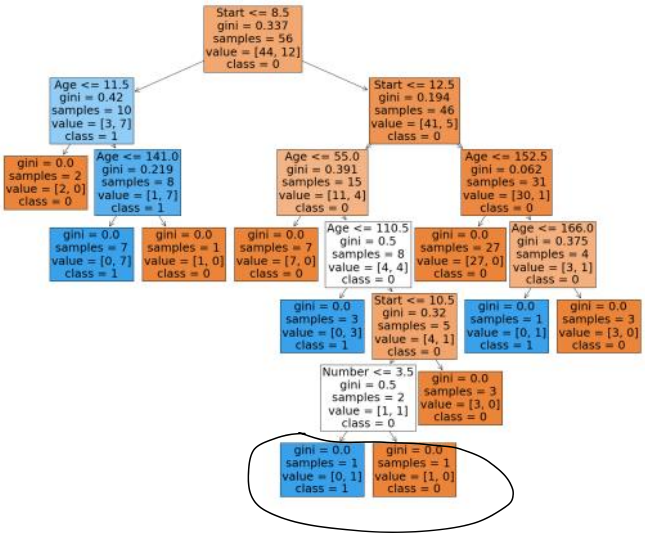
x1	x2	y	P(y=0)	P(y=1)
78	26	0	0.5	0.5
40	20	0	0.5	0.5
3	5	1	0.33	0.67
80	23	0	0.5	0.5

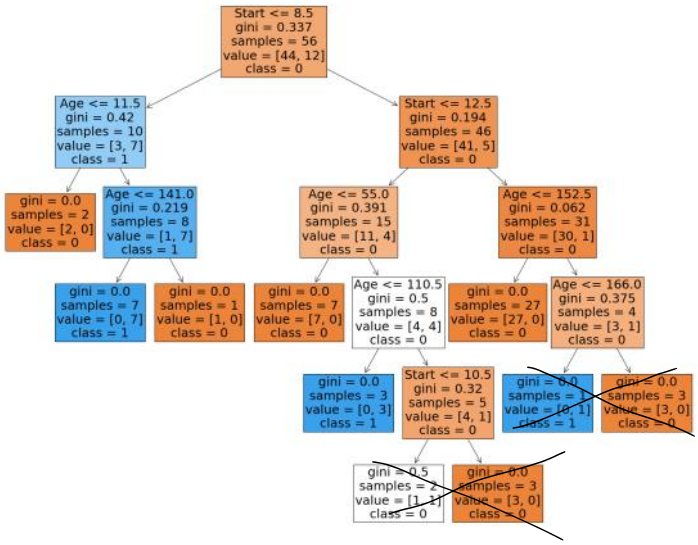
Tree	Accuracy	Remark
	0.25	Over-Fitting

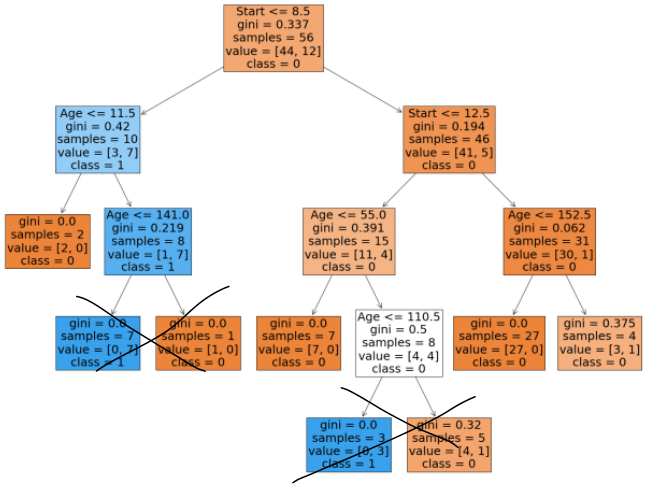
	0.75	Fitting
	0.25	Under-Fitting

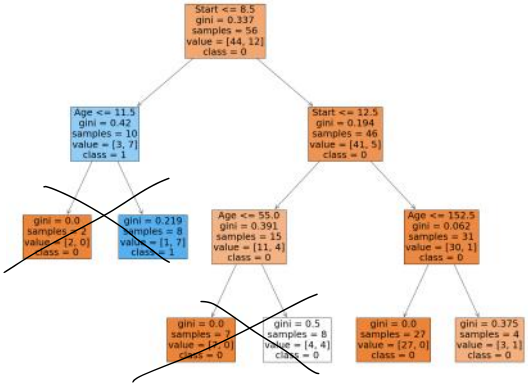
min_samples_split : *int or float, default=2*

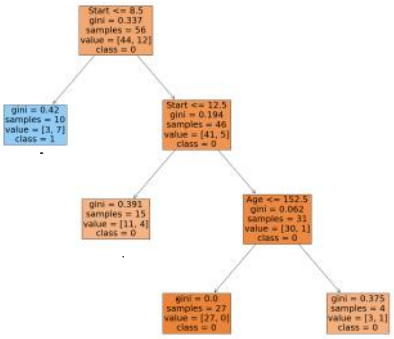
The minimum number of samples required to split an internal node

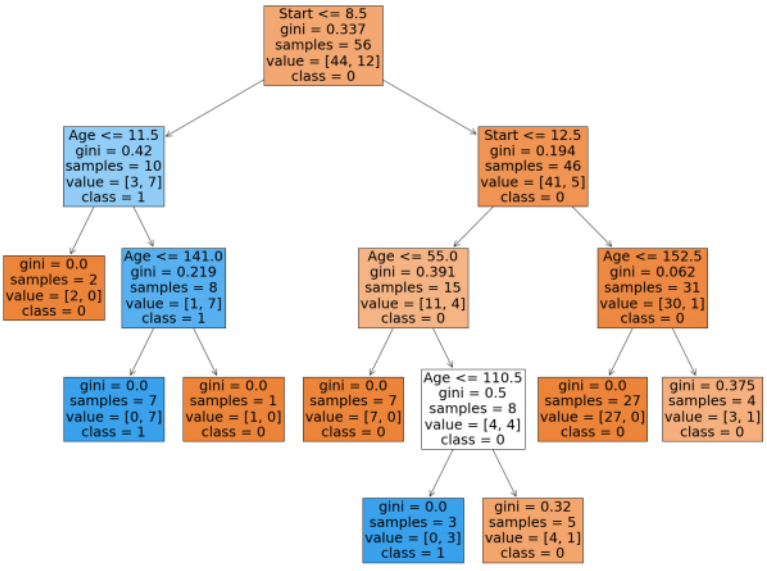


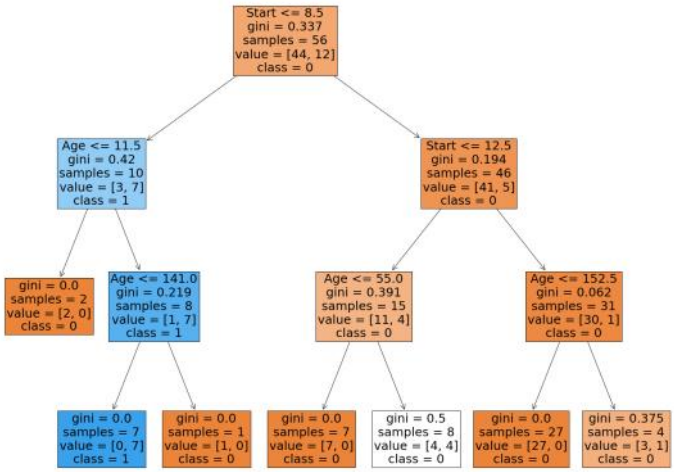



Start <= 8.5
gini = 0.337
samples = 56
value = [44, 12]
class = 0
Age <= 11.5
gini = 0.42
samples = 10
value = [3, 7]
class = 1
Start <= 12.5
gini = 0.194
samples = 46
value = [41, 5]
class = 0
gini = 0.0
samples = 2
value = [2, 0]
class = 0
Age <= 141.0
gini = 0.219
samples = 8
value = [1, 7]
class = 1
Age <= 55.0
gini = 0.391
samples = 15
value = [11, 4]
class = 0
Age <= 152.5
gini = 0.062
samples = 31
value = [30, 1]
class = 0
gini = 0.0
samples = 7
value = [0, 7]
class = 1
gini = 0.0
samples = 1
value = [1, 0]
class = 0
gini = 0.0
samples = 7
value = [7, 0]
class = 0
Age <= 110.5
gini = 0.5
samples = 8
value = [4, 4]
class = 0
gini = 0.0
samples = 27
value = [27, 0]
class = 0
gini = 0.375
samples = 4
value = [3, 1]
class = 0
gini = 0.0
samples = 3
value = [0, 3]
class = 1
gini = 0.32
samples = 5
value = [4, 1]
class = 0


Start <= 8.5
gini = 0.337
samples = 56
value = [44, 12]
class = 0
Age <= 11.5
gini = 0.42
samples = 10
value = [3, 7]
class = 1
Start <= 12.5
gini = 0.194
samples = 46
value = [41, 5]
class = 0
gini = 0.0
samples = 2
value = [2, 0]
class = 0
gini = 0.219
samples = 8
value = [1, 7]
class = 1
Age <= 55.0
gini = 0.391
samples = 15
value = [11, 4]
class = 0
Age <= 152.5
gini = 0.062
samples = 31
value = [30, 1]
class = 0
gini = 0.0
samples = 7
value = [7, 0]
class = 0
gini = 0.5
samples = 8
value = [4, 4]
class = 0
gini = 0.0
samples = 27
value = [27, 0]
class = 0
gini = 0.375
samples = 4
value = [3, 1]
class = 0

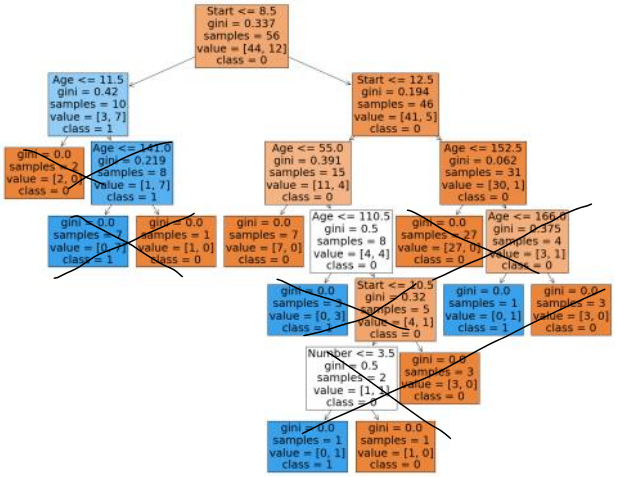

Start <= 8.5
gini = 0.337
samples = 56
value = [44, 12]
class = 0
gini = 0.42
samples = 10
value = [3, 7]
class = 1
Start <= 12.5
gini = 0.194
samples = 46
value = [41, 5]
class = 0
gini = 0.391
samples = 15
value = [11, 4]
class = 0
Age <= 152.5
gini = 0.062
samples = 31
value = [30, 1]
class = 0
gini = 0.0
samples = 27
value = [27, 0]
class = 0
gini = 0.375
samples = 4
value = [3, 1]
class = 0

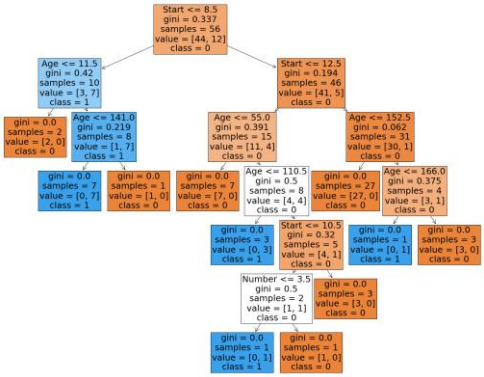

Start <= 8.5
gini = 0.337
samples = 56
value = [44, 12]
class = 0
Age <= 11.5
gini = 0.42
samples = 10
value = [3, 7]
class = 1
Start <= 12.5
gini = 0.194
samples = 46
value = [41, 5]
class = 0
gini = 0.0
samples = 2
value = [2, 0]
class = 0
Age <= 141.0
gini = 0.219
samples = 8
value = [1, 7]
class = 1
Age <= 55.0
gini = 0.391
samples = 15
value = [11, 4]
class = 0
Age <= 152.5
gini = 0.062
samples = 31
value = [30, 1]
class = 0
gini = 0.0
samples = 7
value = [0, 7]
class = 1
gini = 0.0
samples = 1
value = [1, 0]
class = 0
gini = 0.0
samples = 7
value = [7, 0]
class = 0
Age <= 110.5
gini = 0.5
samples = 8
value = [4, 4]
class = 0
gini = 0.0
samples = 27
value = [27, 0]
class = 0
gini = 0.375
samples = 4
value = [3, 1]
class = 0
gini = 0.0
samples = 3
value = [0, 3]
class = 1
gini = 0.32
samples = 5
value = [4, 1]
class = 0

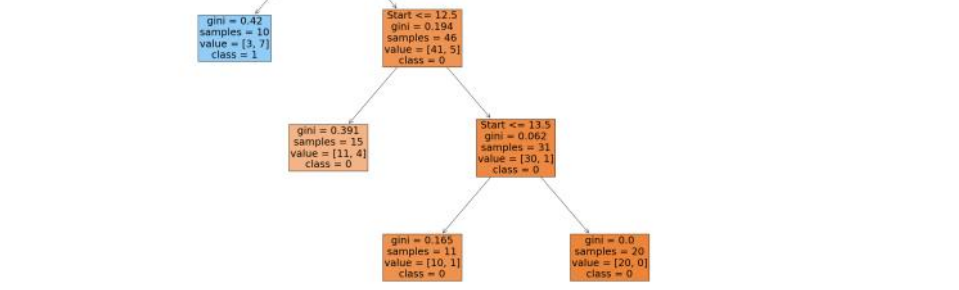


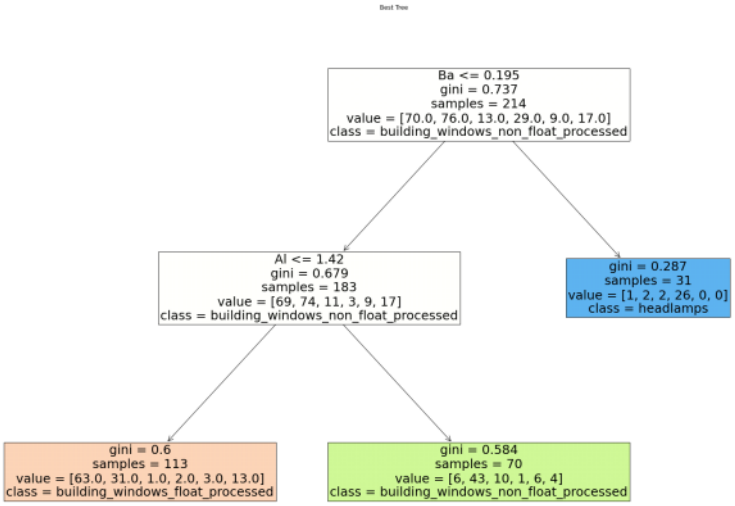
min_samples_leaf : *int or float, default=1*

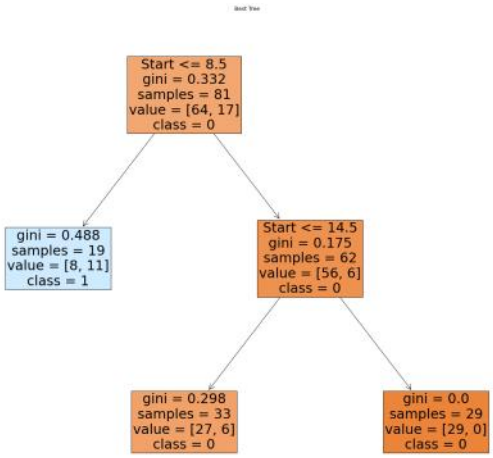
The minimum number of samples required to be at a leaf node.



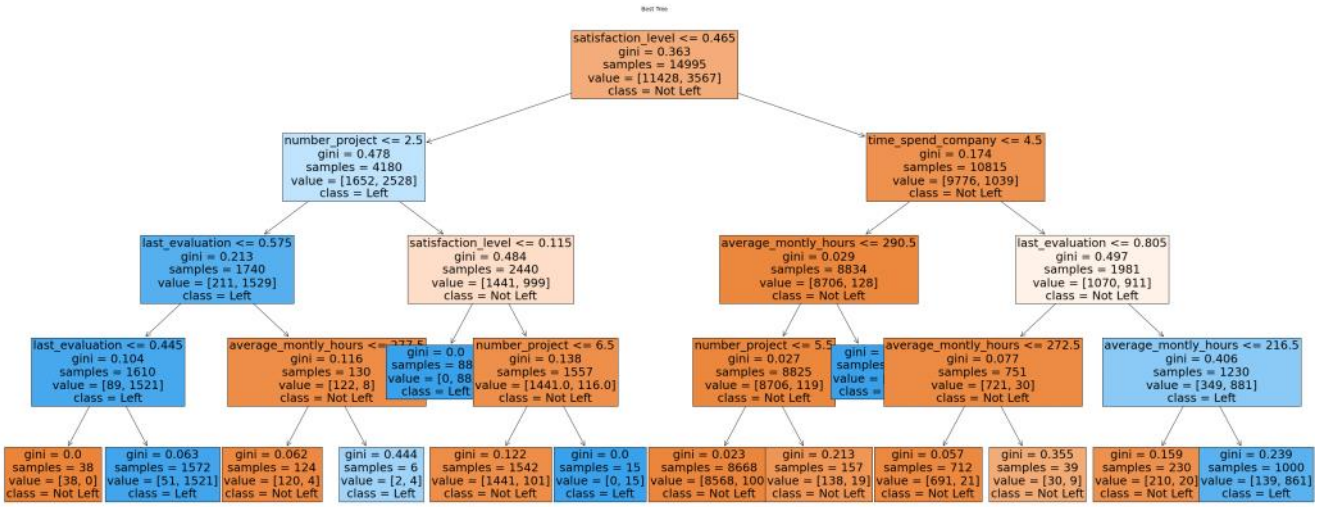


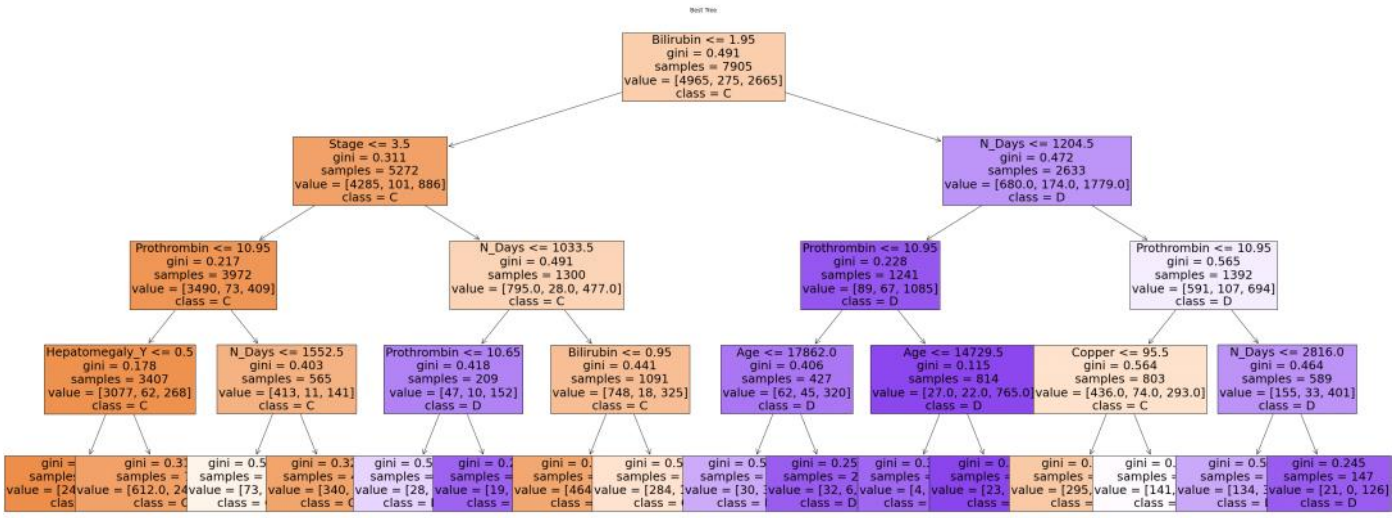


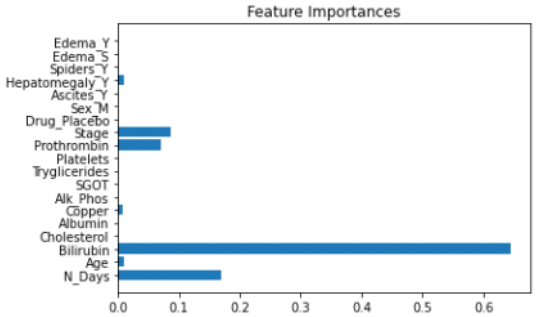

Ba <= 0.195
gini = 0.737
samples = 214
value = [70.0, 76.0, 13.0, 29.0, 9.0, 17.0]
class = building_windows_non_float_processed
Al <= 1.42
gini = 0.679
samples = 183
value = [69, 74, 11, 3, 9, 17]
class = building_windows_non_float_processed
gini = 0.287
samples = 31
value = [1, 2, 2, 26, 0, 0]
class = headlamps
gini = 0.6
samples = 113
value = [63.0, 31.0, 1.0, 2.0, 3.0, 13.0]
class = building_windows_float_processed
gini = 0.584
samples = 70
value = [6, 43, 10, 1, 6, 4]
class = building_windows_non_float_processed

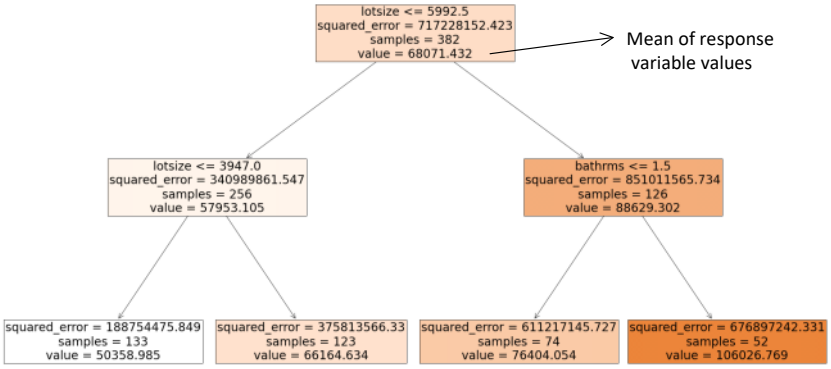

Start <= 8.5
gini = 0.332
samples = 81
value = [64, 17]
class = 0
gini = 0.488
samples = 19
value = [8, 11]
class = 1
Start <= 14.5
gini = 0.175
samples = 62
value = [56, 6]
class = 0
gini = 0.298
samples = 33
value = [27, 6]
class = 0
gini = 0.0
samples = 29
value = [29, 0]
class = 0

Variable Importance: The average reduction in impurity contributed by the variable

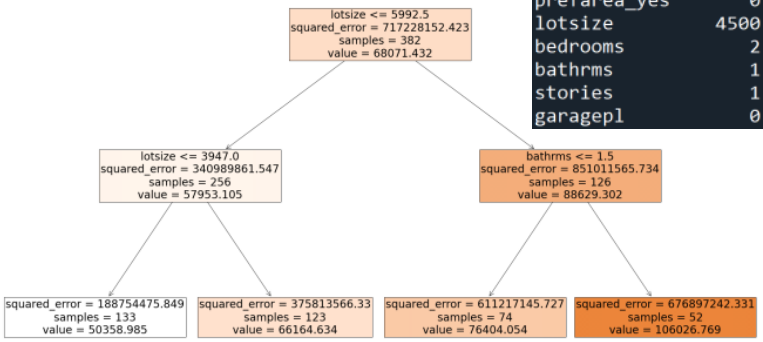








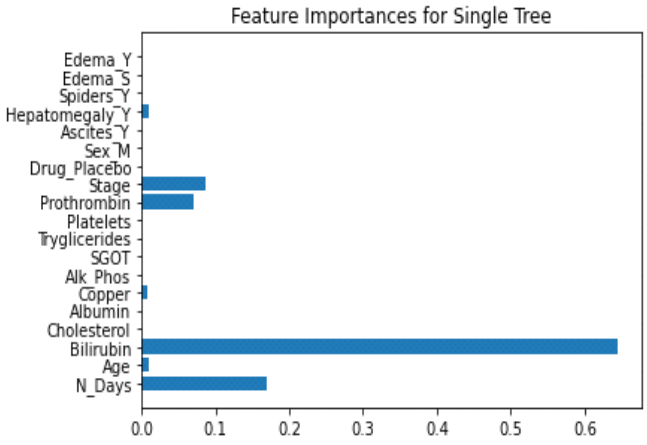
driveway_yes	1.0
recroom_yes	0.0
fullbase_yes	0.0
gashw_yes	0.0
airco_yes	0.0
prefarea_yes	0.0
lotsize	4500.0
bedrooms	2.0
bathrms	1.0
stories	1.0
garagepl	0.0

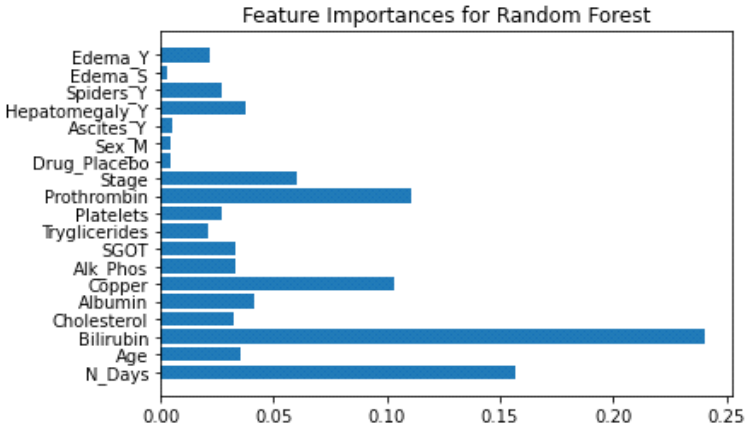


Random Forest

04 May 2024 15:56

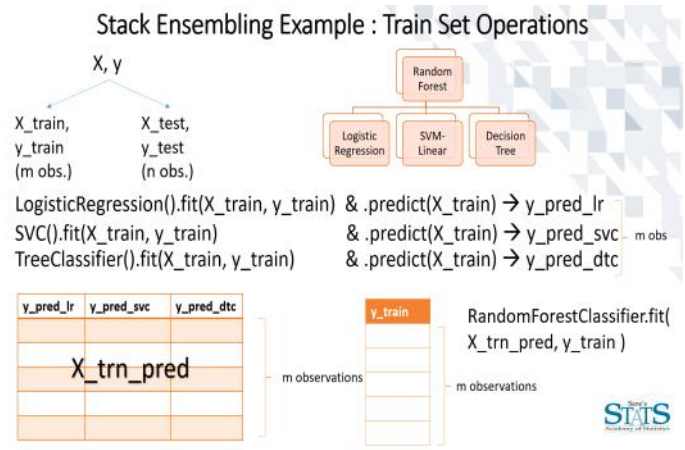
Score: -0.5332237830893267

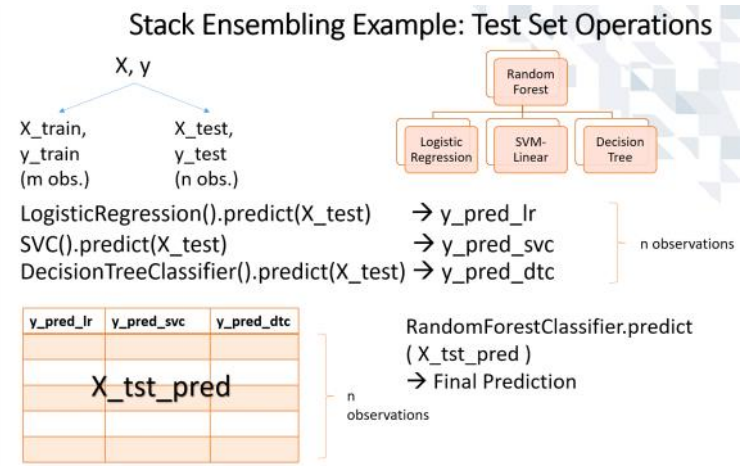




Stacking

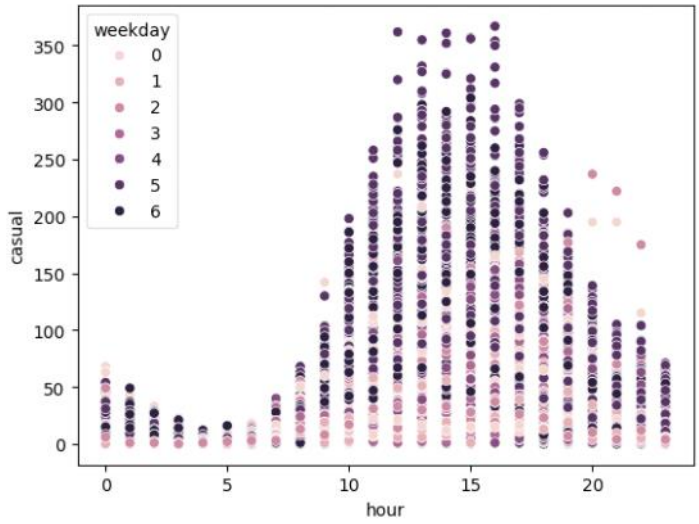
06 May 2024 14:14

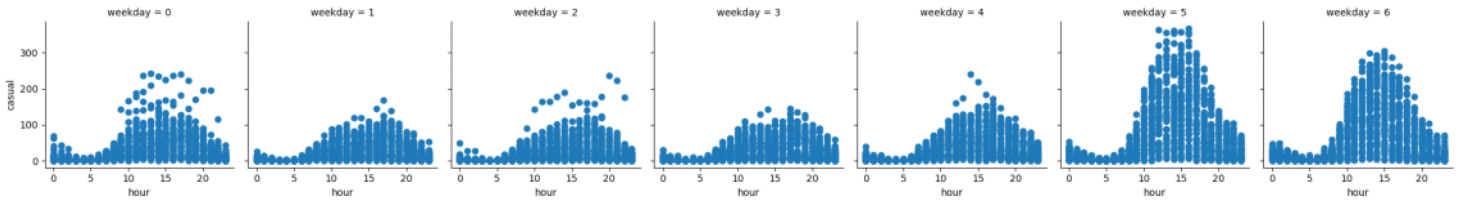


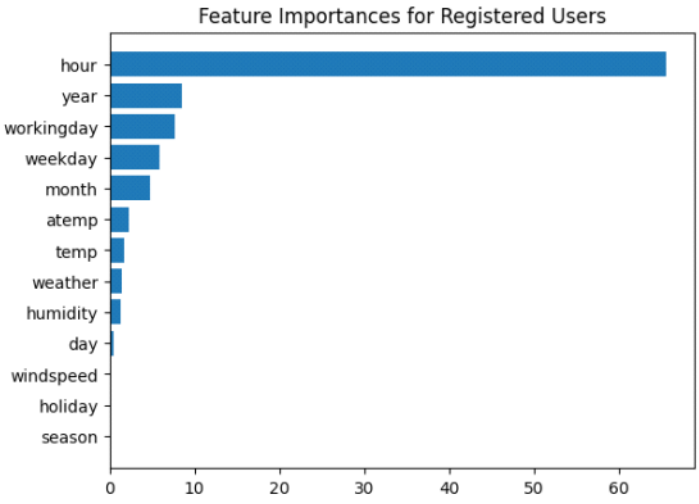


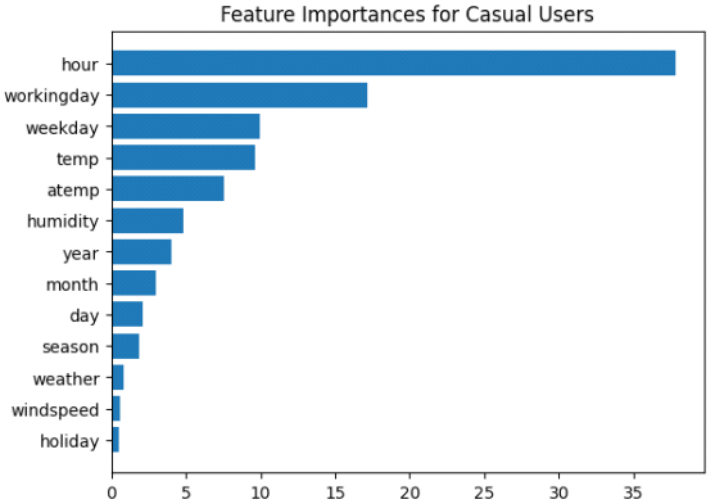
EDA

07 May 2024 10:16



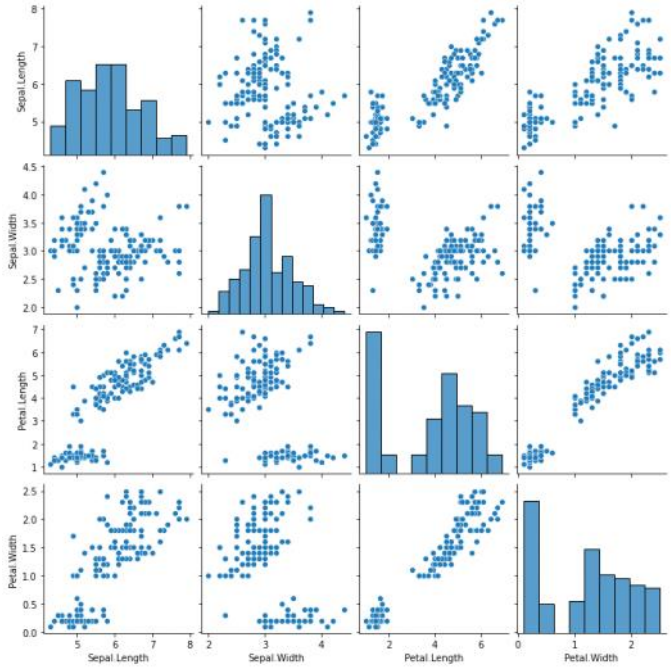


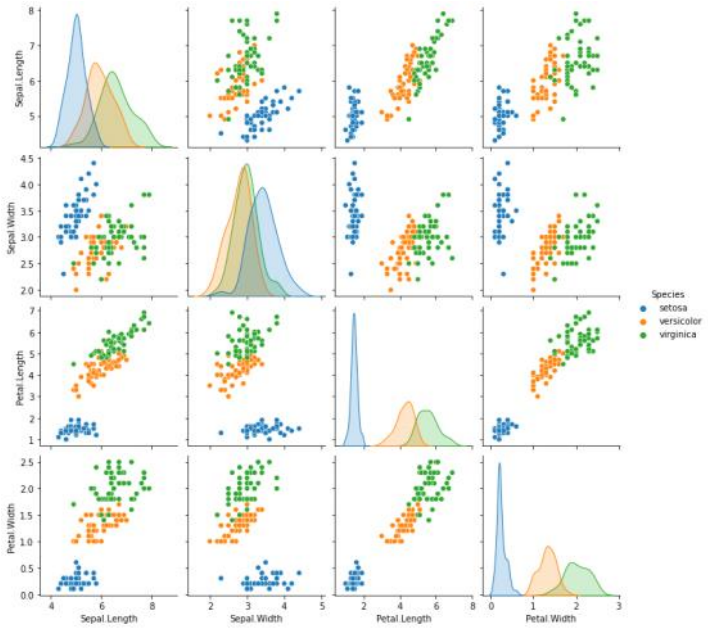


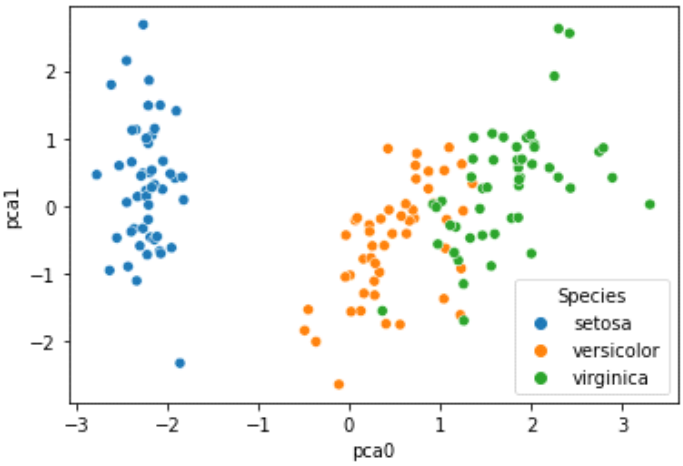


PCA

07 May 2024 11:44

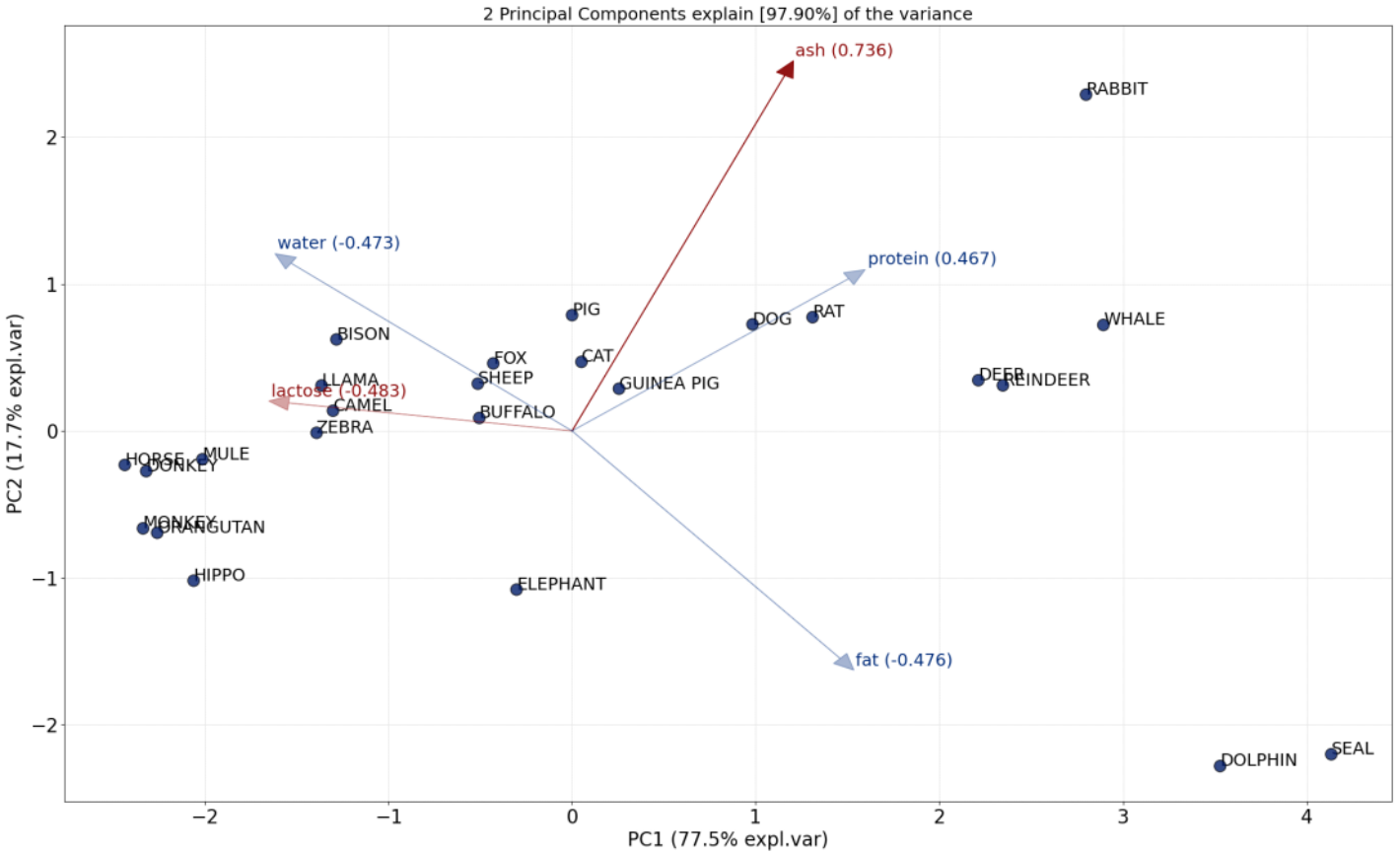


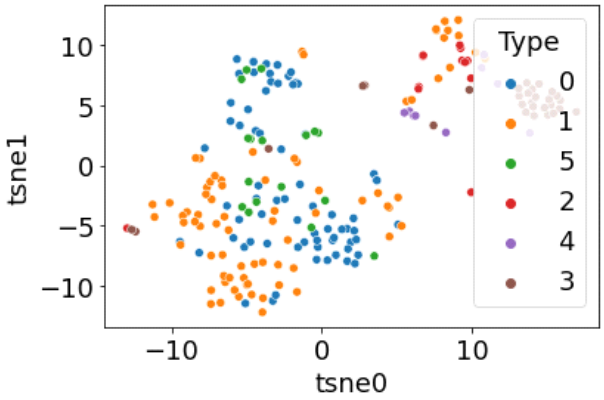


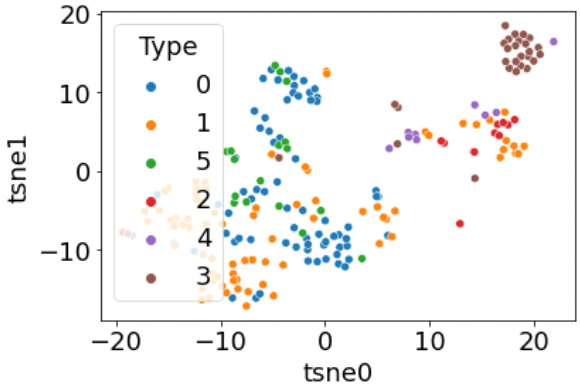


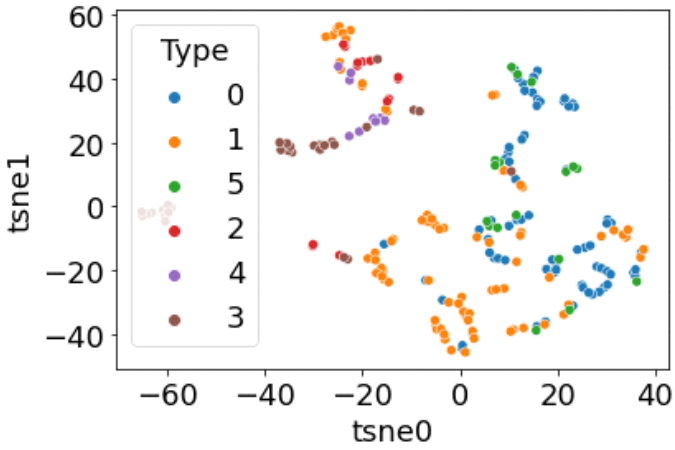
Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
...	...	...	...	...
6.7	3.0	5.2	2.3	virginica
6.3	2.5	5.0	1.9	virginica
6.5	3.0	5.2	2.0	virginica
6.2	3.4	5.4	2.3	virginica
5.9	3.0	5.1	1.8	virginica

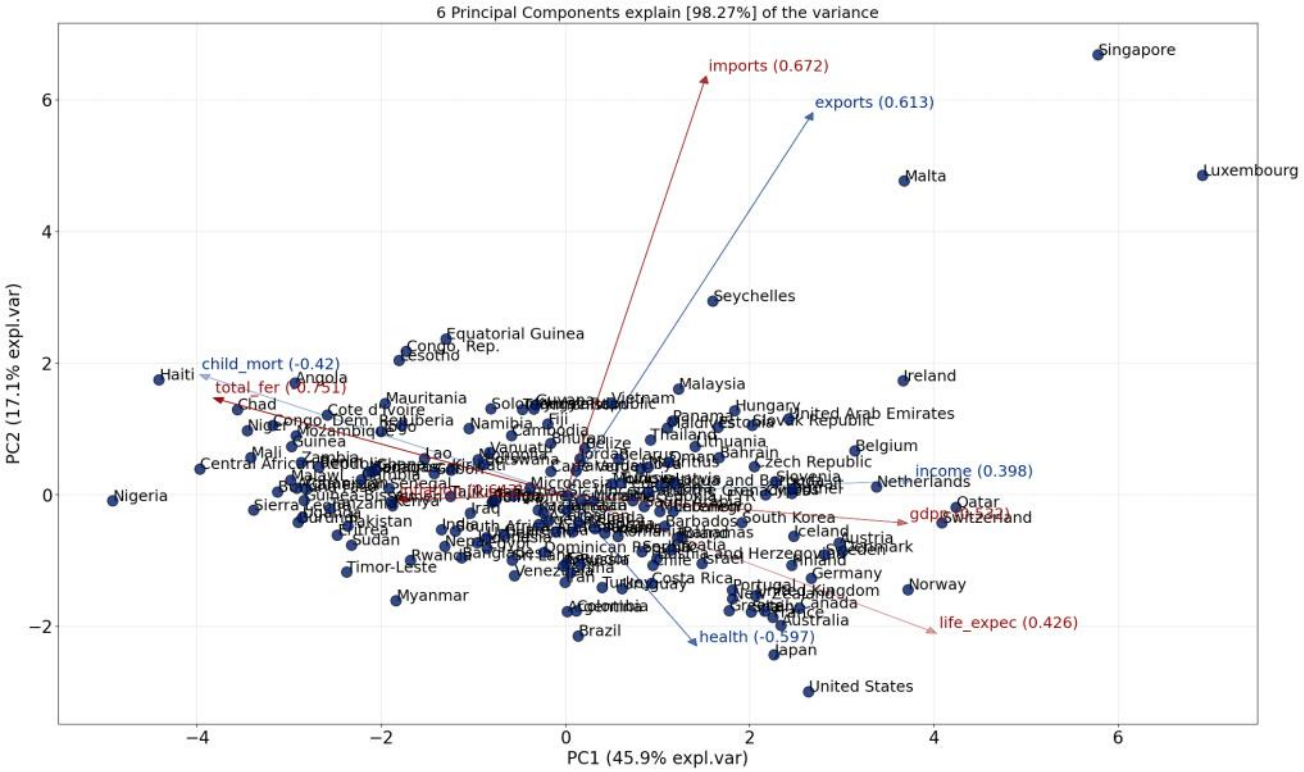
	pca0	pca1	pca2	pca3	Species
	-2.264703	0.480027	-0.127706	-0.024168	setosa
	-2.080961	-0.674134	-0.234609	-0.103007	setosa
	-2.364229	-0.341908	0.044201	-0.028377	setosa
	-2.299384	-0.597395	0.091290	0.065956	setosa
	-2.389842	0.646835	0.015738	0.035923	setosa
	...	...	...	...	...
	1.870503	0.386966	0.256274	-0.389257	virginica
	1.564580	-0.896687	-0.026371	-0.220192	virginica
	1.521170	0.269069	0.180178	-0.119171	virginica
	1.372788	1.011254	0.933395	-0.026129	virginica
	0.960656	-0.024332	0.528249	0.163078	virginica


2 Principal Components explain [97.90%] of the variance
ash (0.736)
water (-0.473)
protein (0.467)
lactose (-0.483)
fat (-0.476)
RABBIT
PIG
DOG
RAT
WHALE
BISON
FOX
CAT
SHEEP
LLAMA
GUINEA PIG
DEER
REINDEER
CAMEL
BUFFALO
ZEBRA
HORSE
MULE
DONKEY
MONKEY
ORANGUTAN
HIPPO
ELEPHANT
DOLPHIN
SEAL
PC1 (77.5% expl.var)
PC2 (17.7% expl.var)


Type
0
1
5
2
4
3
tsne0
tsne1

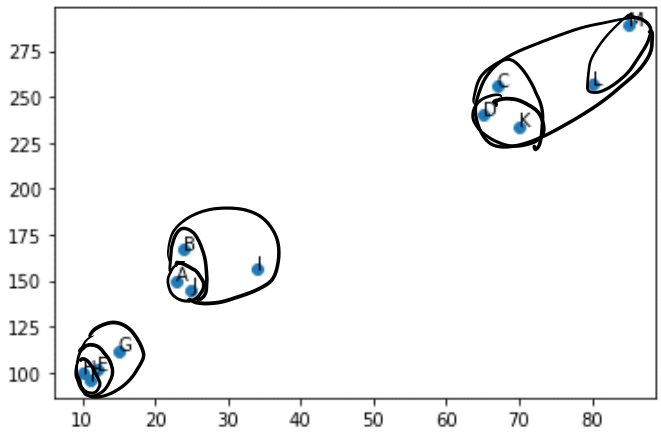

Type
0
1
5
2
4
3
tsne0
tsne1

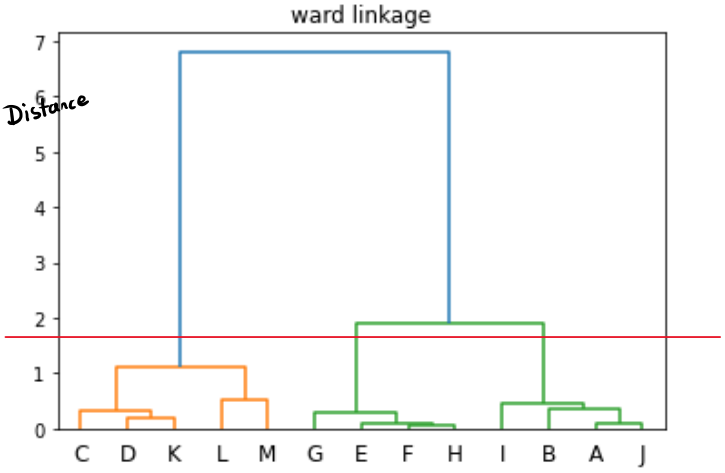

Type
0
1
5
2
4
3
tsne1
tsne0
60
40
20
0
−20
−40
−60
−40
−20
0
20
40

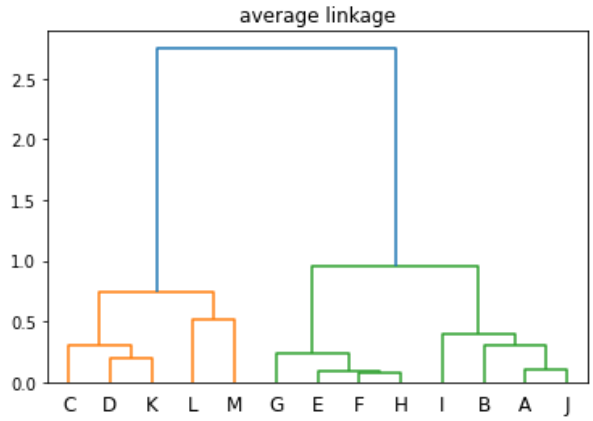

6 Principal Components explain [98.27%] of the variance
imports (0.672)
exports (0.613)
child_mort (-0.42)
total_fer (0.751)
income (0.398)
gdpp (0.332)
life_expec (0.426)
health (-0.597)
Singapore
Luxembourg
Malta
Seychelles
Equatorial Guinea
Haiti
Angola
Ireland
Nigeria
Brazil
Japan
United States
Myanmar
Timor-Leste
Norway
Qatar
Switzerland
Netherlands
Belgium
Czech Republic
United Arab Emirates
Malaysia
Hungary
Iceland
Germany
Australia
Canada
Portugal
Costa Rica
South Korea
PC1 (45.9% expl.var)
PC2 (17.1% expl.var)

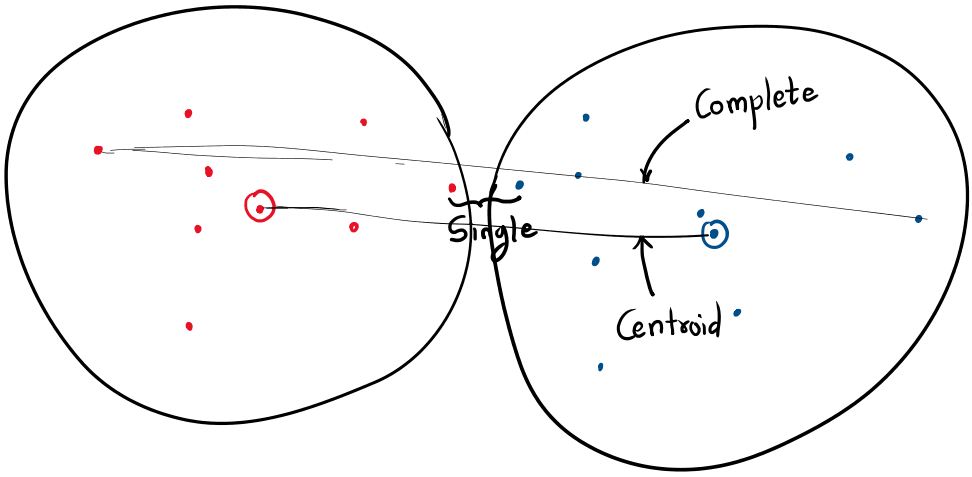
Clustering

06 April 2024 14:06



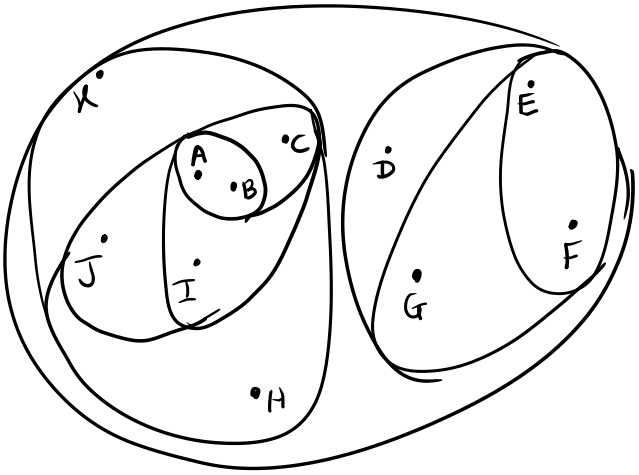


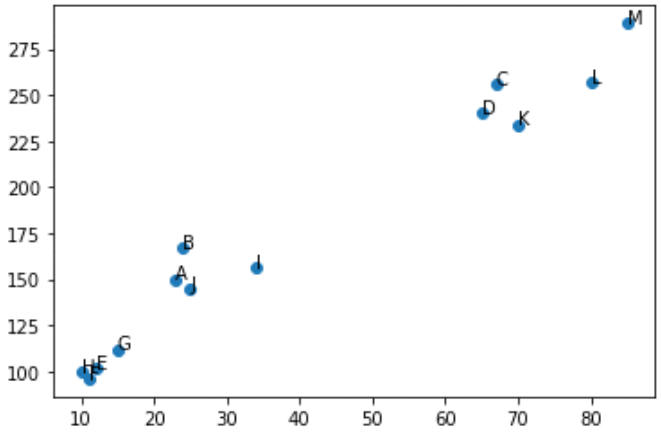




Ward : based on ANOVA

Average



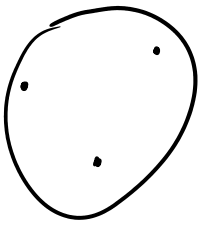


$$\text{Sil coef} = \frac{b-a}{\max(a,b)}$$

b: avg nearest cluster distance

a: avg intra-cluster distance

Sil score = avg (Sil coef of all point)



Datasets / milk.csv

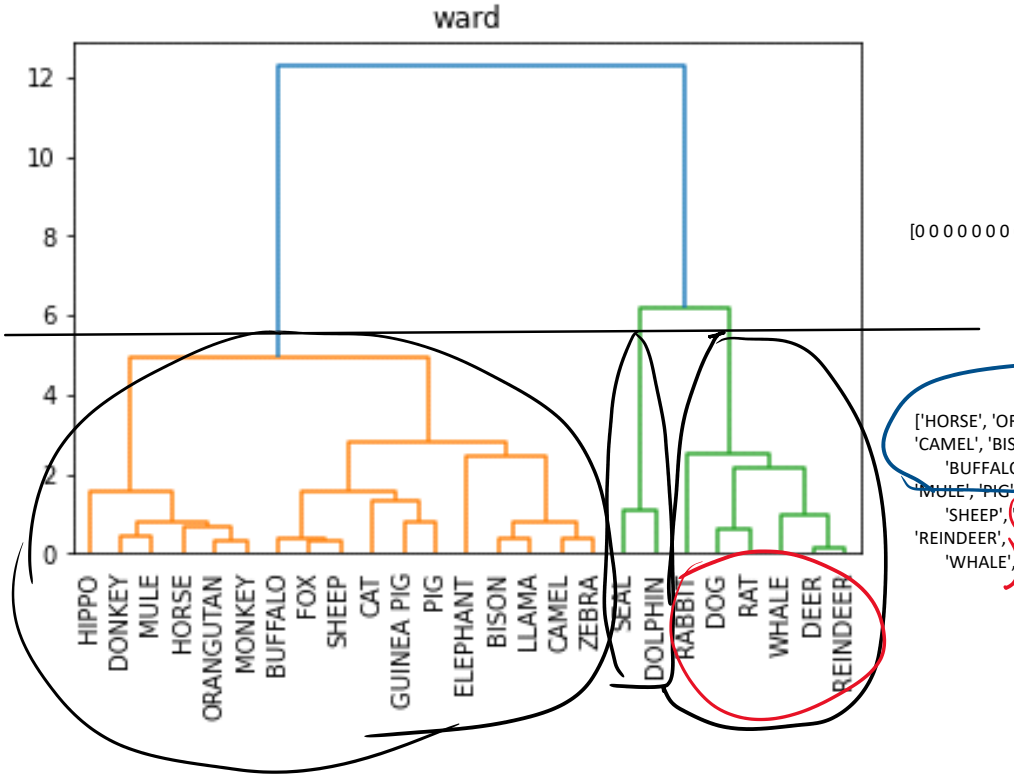
index_col = 0

1) Dendrogram

2) To... clusters = 2 3 4 5 6

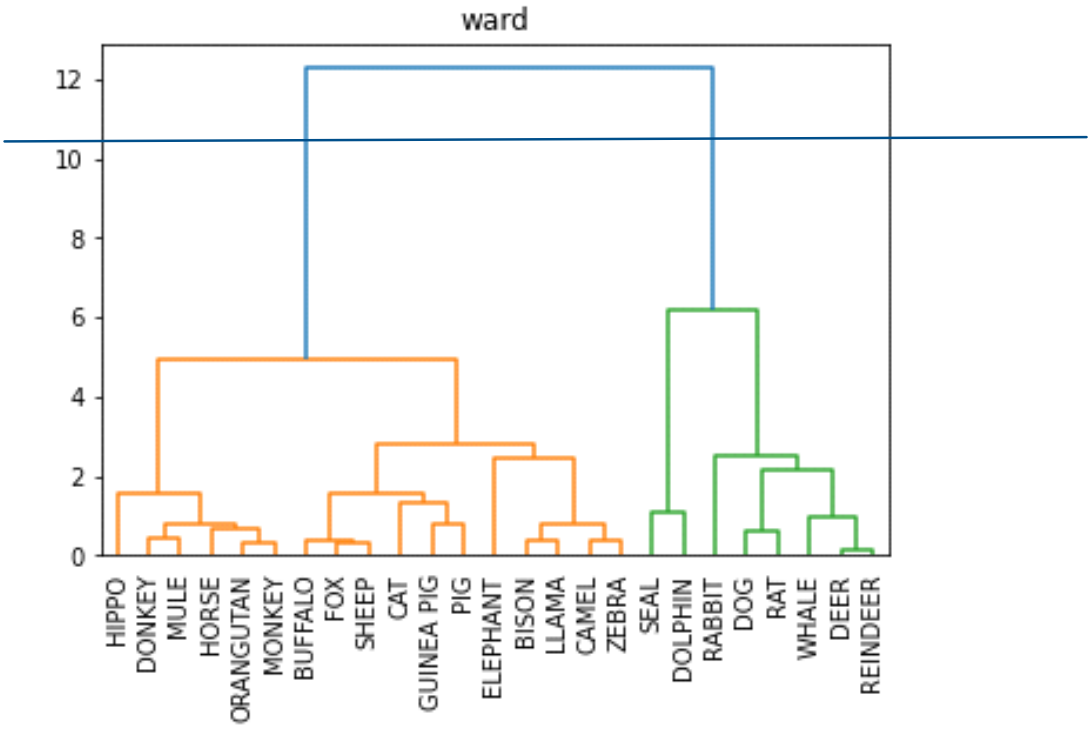
1) Dendrogram

2) Try clusters = 2, 3, 4, 5, 6



[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 2 2]

['HORSE', 'ORANGUTAN', 'MONKEY', 'DONKEY', 'HIPPO', 'CAMEL', 'BISON',
'BUFFALO', 'GUINEA PIG', 'CAT', 'FOX', 'LLAMA', 'MULE', 'PIG', 'ZEBRA',
'SHEEP', 'DOG', 'ELEPHANT', 'RABBIT', 'RAT', 'DEER', 'REINDEER',
'WHALE', 'SEAL', 'DOLPHIN']



Datasets / nutrient.csv

index_col = 0

1) Dendrogram

2) Try clusters = 2, 3, 4, 5, 6

Datasets / Protein.csv

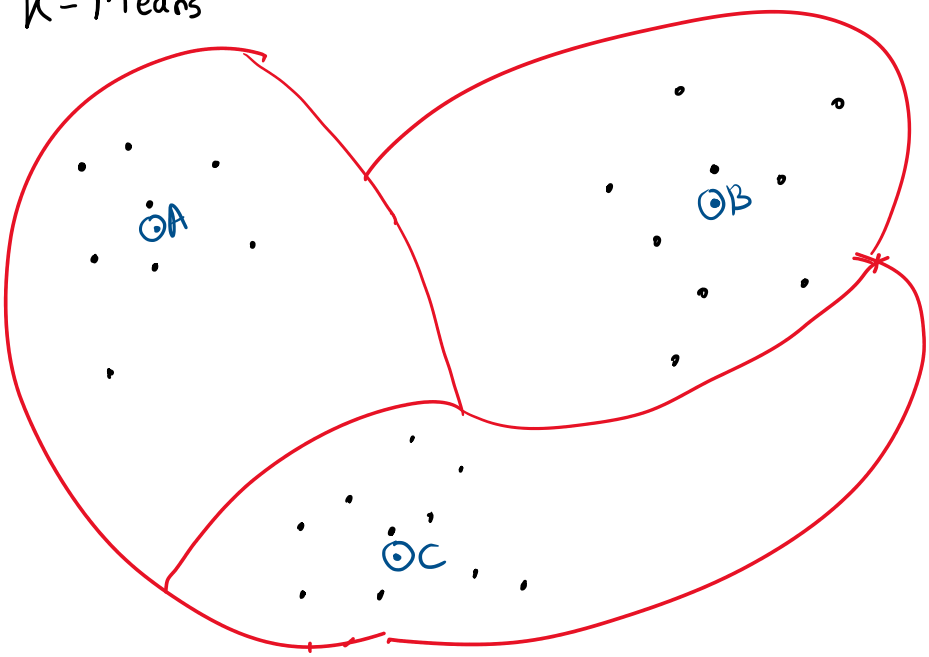
index_col = 0

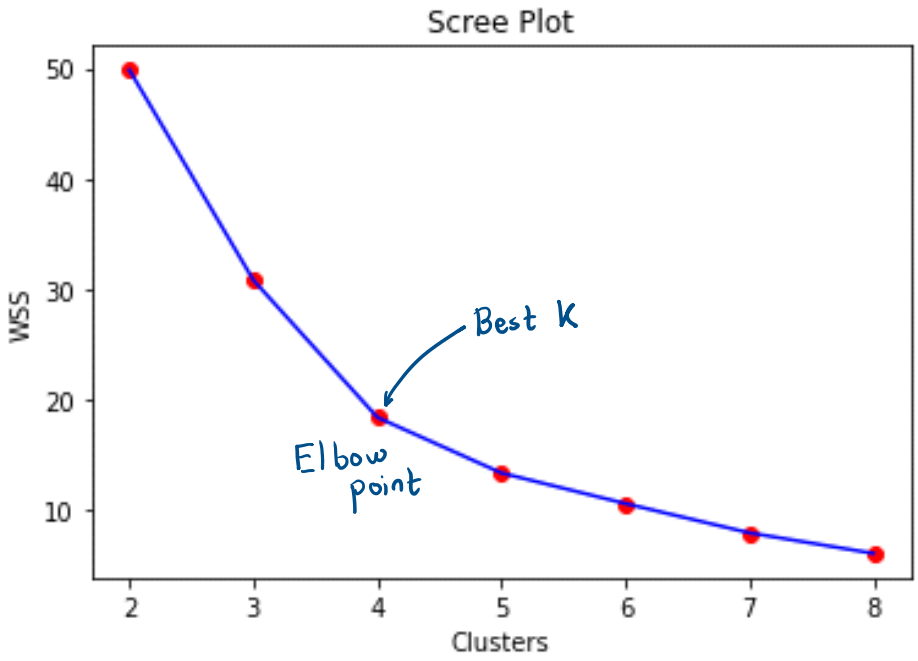
1) Dendrogram

2) Try clusters = 2, 3, 4, 5, 6

$x_1\ y_1$ $x_2\ y_2$ $x_4\ y_4$ $x_3\ y_3$ C $\left(\frac{x_1+x_2+x_3+x_4}{4}, \frac{y_1+y_2+y_3+y_4}{4}\right)$

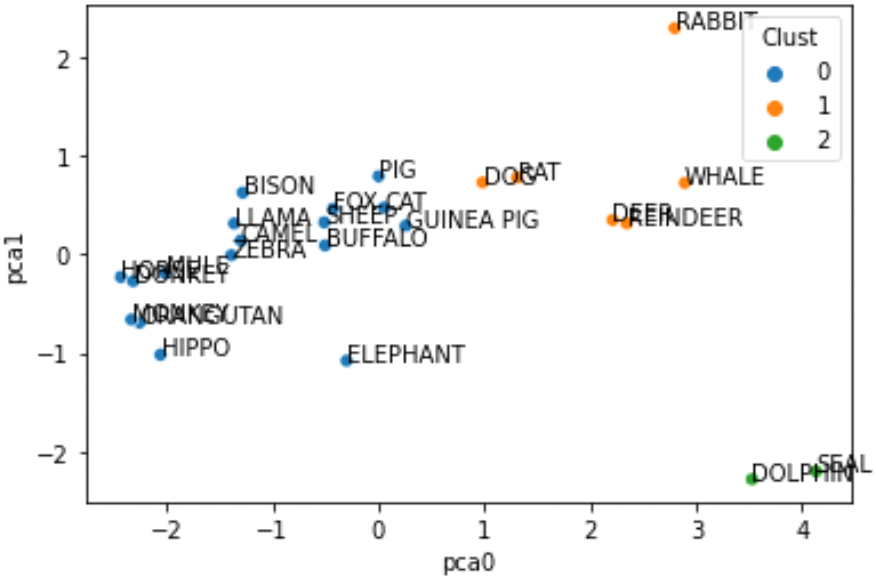
K-Means





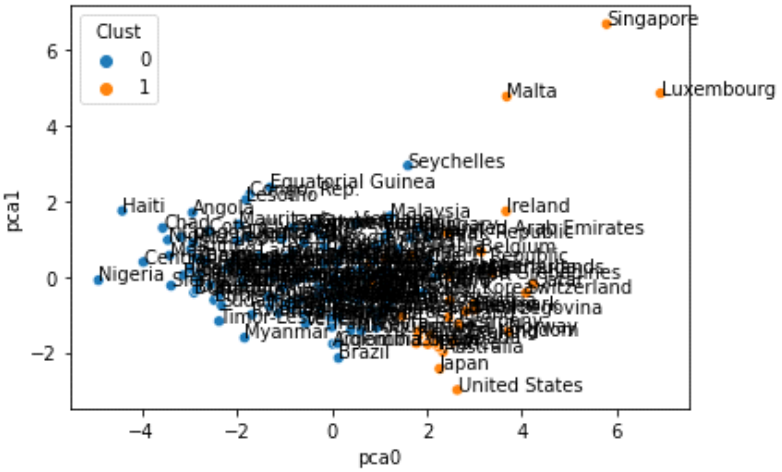
Name	X1	X2
A	-0.620863	-0.411108
B	-0.584506	-0.154456
C	0.978838	1.189194
D	0.906124	0.947639
E	-1.020788	-1.135773
F	-1.057145	-1.226356
G	-0.911718	-0.984801
H	-1.093502	-1.165967
I	-0.220938	-0.320525
J	-0.548149	-0.486594
K	1.087909	0.857056
L	1.451477	1.204291
M	1.633261	1.687401

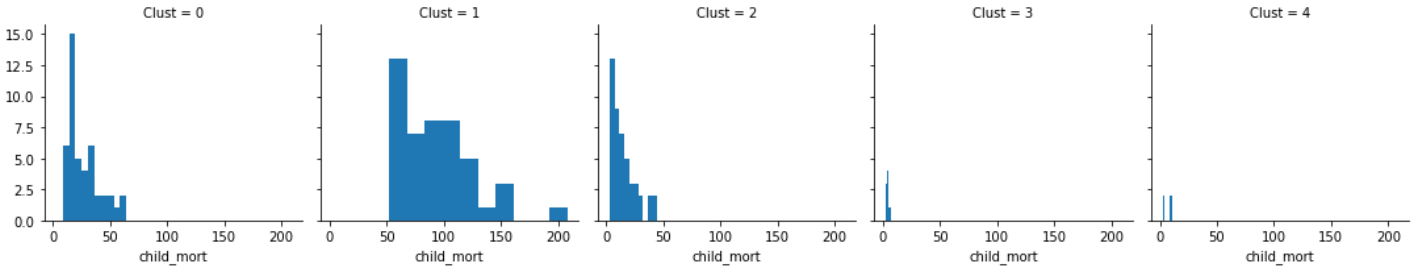
```
In [15]: print(clust.labels_)
[0 0 1 1 0 0 0 0 0 0 1 1 1]
```



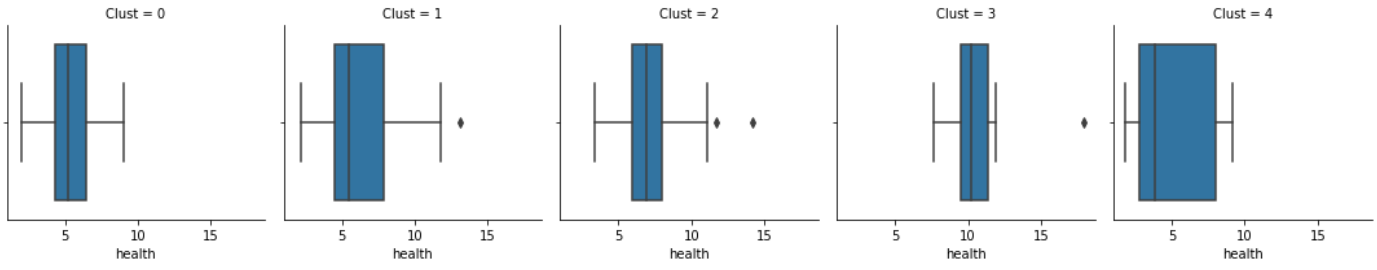
Ward's Distance

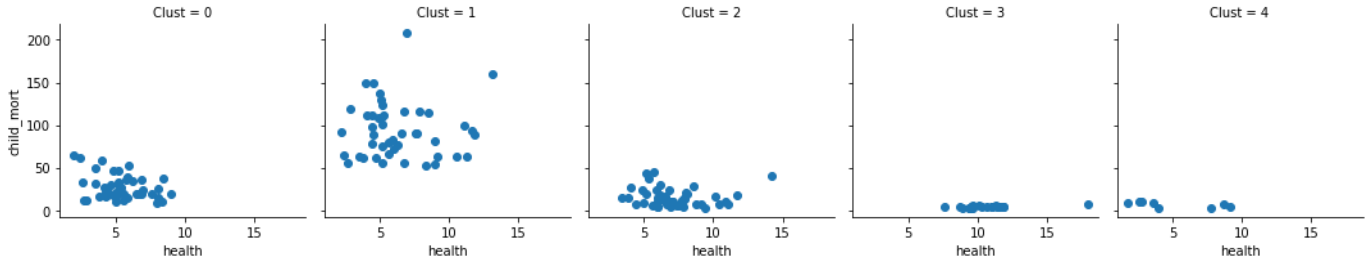
https://www.statisticshowto.com/wards-method/#:~:text=Calculate%20the%20distance%20between%20each,of%20squares%20from%20Step%204.

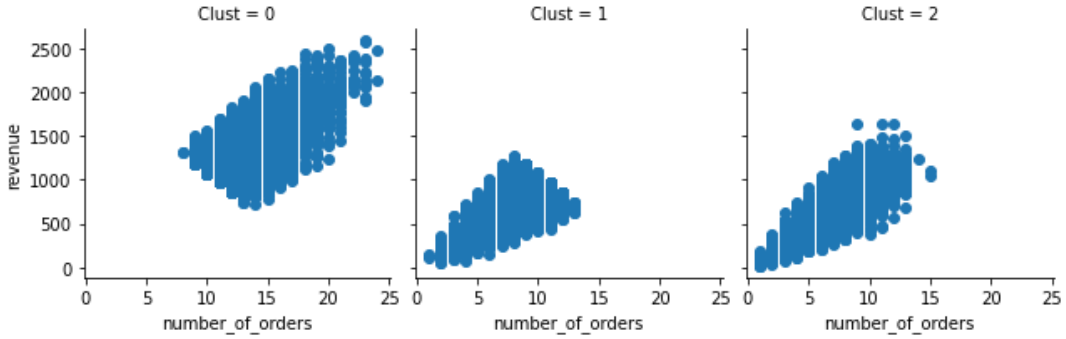


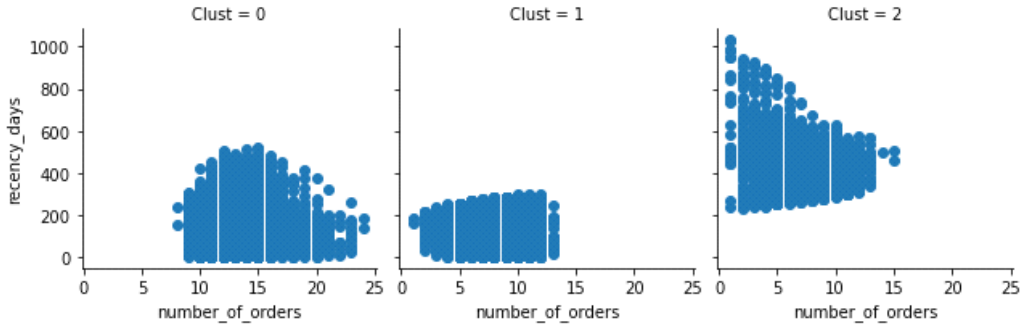


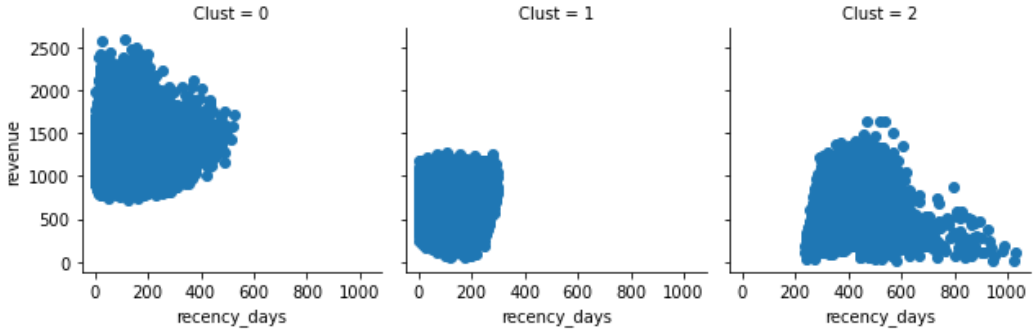

```
In [97]: clust_data.groupby('Clust')['child_mort'].mean()
Out[97]:
Clust
0    26.917778
1    94.180435
2    15.160870
3     4.300000
4     6.937500
```

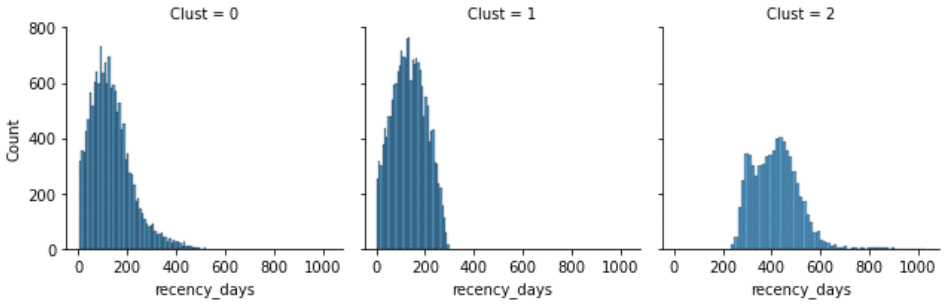




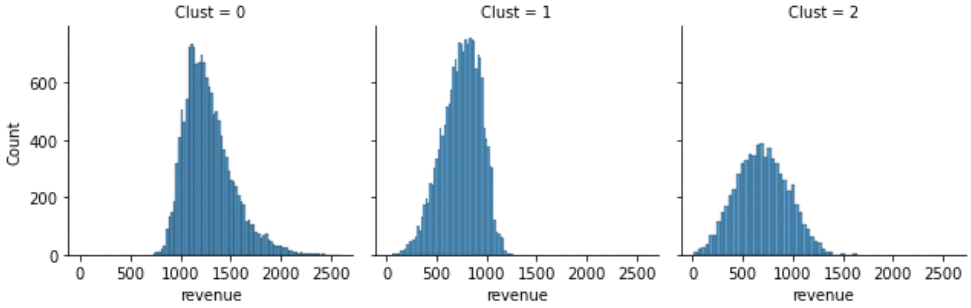

Clust = 0
Clust = 1
Clust = 2
revenue
number_of_orders

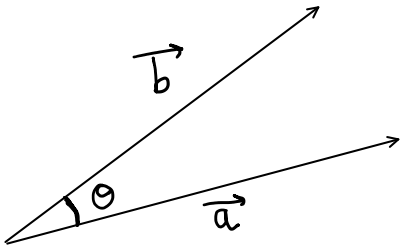

Clust = 0
Clust = 1
Clust = 2
recency_days
number_of_orders


Clust = 0
Clust = 1
Clust = 2
revenue
recency_days


Clust = 0
Clust = 1
Clust = 2
Count
recency_days


Clust = 0
Clust = 1
Clust = 2
Count
number_of_orders


Clust = 0
Clust = 1
Clust = 2
Count
0
200
400
600
0
500
1000
1500
2000
2500
revenue



$\cos 90^\circ = 0$

$\cos 0^\circ = 1$

$\therefore$ cosine $\cos\theta = \frac{\text{Adjacent side}}{\text{Hypotenuse}}$

$|\vec{a}|$: length of $\vec{a}$

$\vec{a} \cdot \vec{b} = |\vec{a}|\,|\vec{b}| \cos\theta$

$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}|\,|\vec{b}|}$

	Product A	Product B	Product C
User 1	4	3	1
User 2	2	2	5
User 3	5	4	1
User 4	2	3	4

Cosine Similarity User(1,2)

User1 = [4, 3, 1]

User2 = [2, 2, 5]

$$= \frac{4(2)+3(2)+1(5)}{\sqrt{4^2+3^2+1^2}\ \sqrt{2^2+2^2+5^2}}$$

Cosine User(1,3)

$$= \frac{4(5)+3(4)+1(1)}{\sqrt{4^2+3^2+1^2}\ \sqrt{5^2+4^2+1^2}} = \frac{19}{\sqrt{26}\ \sqrt{33}} = 0.648649304$$

$$= \frac{33}{\sqrt{26}\ \sqrt{42}}$$

$$= 0.998$$

Sno	A	B	C
1	97	93	99
2	73	14	94
3	93	93	87
4	100	55	66
5	23	77	59

Sno	variable	value
1	A	97
2	A	73
3	A	93
4	A	100
5	A	23
1	B	93
2	B	14
3	B	93
4	B	55
5	B	77
1	C	99
2	C	94
3	C	87
4	C	66
5	C	59

Minimum Project Requirements

10 May 2024 15:51

1. Should be a problem which has some commercial value
2. Should cover usage of as much as possible topics covered in the syllabus
3. ML project types:
    a. Structured Data
    b. Text Data
    c. Image Data
4. Possible Tasks:
    a. Classification
    b. Regression
    c. Object detection (in case of image data)
    d. Questions & Answers (NLP / LLM)
5. Domains:
    a. Health-care
    b. E-commerce
    c. Banking & Finance
    d. City-related problems
    e. Education
    f. Hospitality

Box-Cox Transformation for positive values

$$y(\lambda) = \begin{cases} \frac{y^{\lambda}-1}{\lambda}, & \text{if } \lambda \neq 0; \\ \log y, & \text{if } \lambda = 0. \end{cases}$$

Box-Cox Transformation for negative values

$$y(\boldsymbol{\lambda}) = \begin{cases} \frac{(y+\lambda_2)^{\lambda_1}-1}{\lambda_1}, & \text{if } \lambda_1 \neq 0; \\ \log(y+\lambda_2), & \text{if } \lambda_1 = 0. \end{cases}$$

What is a Box Cox Transformation?

A Box Cox transformation is a transformation of non-normal dependent variables into a normal shape. Normality is an important assumption for many statistical techniques; if your data isn't normal, applying a Box-Cox means that you are able to run a broader number of tests.

From <https://www.statisticshowto.com/probability-and-statistics/normal-distributions/box-cox-transformation/>

Bigmart

14 May 2024 10:43

Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
OUT049	1999	Medium	Tier 1	Supermarket Type1
OUT018	2009	Medium	Tier 3	Supermarket Type2
OUT010	1998	nan	Tier 3	Grocery Store
OUT013	1987	High	Tier 3	Supermarket Type1
OUT027	1985	Medium	Tier 3	Supermarket Type3
OUT045	2002	nan	Tier 2	Supermarket Type1
OUT017	2007	nan	Tier 2	Supermarket Type1
OUT046	1997	Small	Tier 1	Supermarket Type1
OUT035	2004	Small	Tier 2	Supermarket Type1
OUT019	1985	Small	Tier 1	Grocery Store