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

```
df = pd.read_csv('/content/WWildBlueberryPollinationSimulationData.csv')
```

```
df.head()
```

	Row#	clonesize	MaxOfLowerTRange	MinOfLowerTRange	AverageOfLowerTRange	R
0	0	37.5	62.0	30.0	50.8	
1	1	37.5	62.0	30.0	50.8	
2	2	37.5	68.2	33.0	55.9	
3	3	37.5	68.2	33.0	55.9	
4	4	37.5	62.0	30.0	50.8	

```
df.shape
```

```
(777, 11)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 777 entries, 0 to 776
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Row#                  777 non-null    int64  
 1   clonesize             777 non-null    float64
 2   MaxOfLowerTRange      777 non-null    float64
 3   MinOfLowerTRange      777 non-null    float64
 4   AverageOfLowerTRange  777 non-null    float64
 5   RainingDays           777 non-null    float64
 6   AverageRainingDays    777 non-null    float64
 7   fruitset              777 non-null    float64
 8   fruitmass             777 non-null    float64
 9   seeds                 777 non-null    float64
 10  yield                 777 non-null    float64
dtypes: float64(10), int64(1)
memory usage: 66.9 KB
```

```
df.isnull().sum()
```

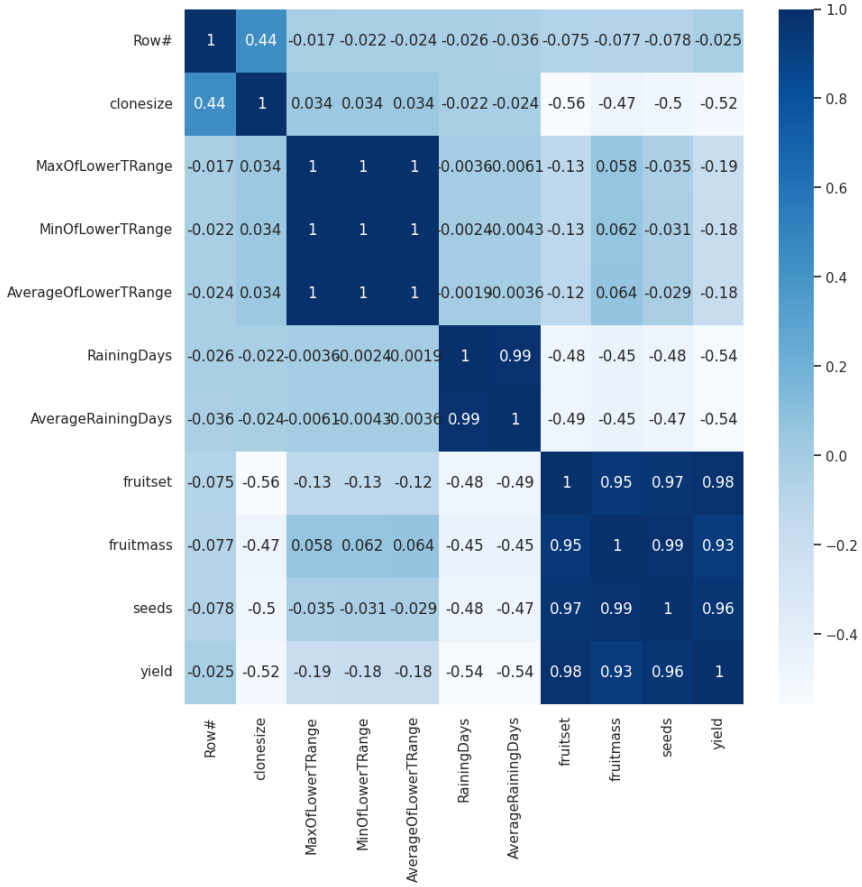
```
Row#                    0
clonesize               0
MaxOfLowerTRange        0
MinOfLowerTRange        0
AverageOfLowerTRange    0
RainingDays             0
AverageRainingDays      0
fruitset                0
fruitmass               0
seeds                   0
yield                   0
dtype: int64
```

```
df.corr()
```

	Row#	clonesize	MaxOfLowerTRange	MinOfLowerTRange	/
Row#	1.000000	0.438706	-0.016947	-0.022430	
clonesize	0.438706	1.000000	0.034295	0.033768	

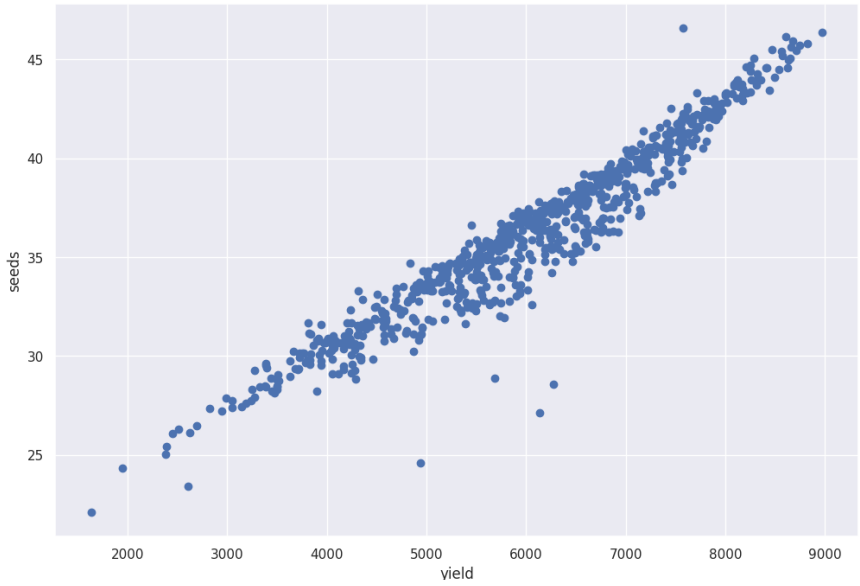
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True, cmap='Blues')
```

```
<Axes: >
```



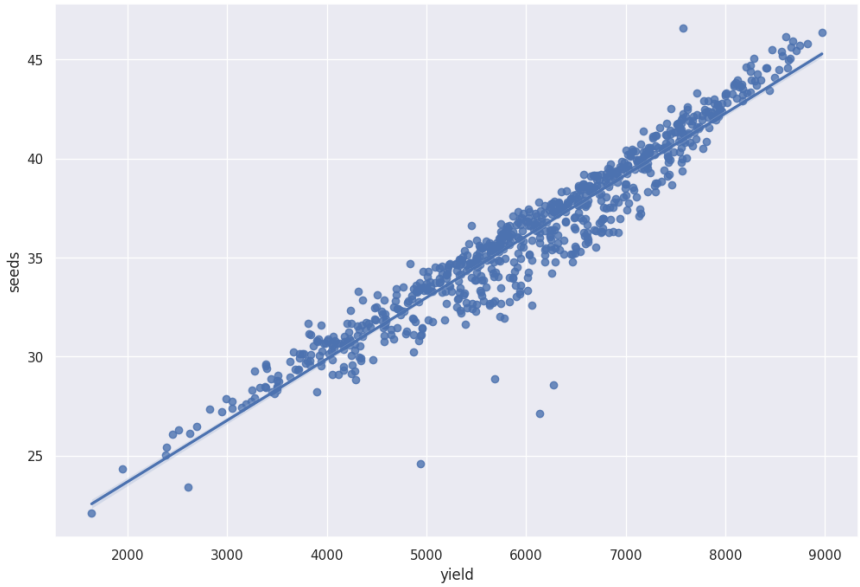

```
plt.subplots(figsize=(12,8))
plt.scatter(df["yield"],df["seeds"])
plt.xlabel("yield")
plt.ylabel("seeds")
```

```
Text(0, 0.5, 'seeds')
```



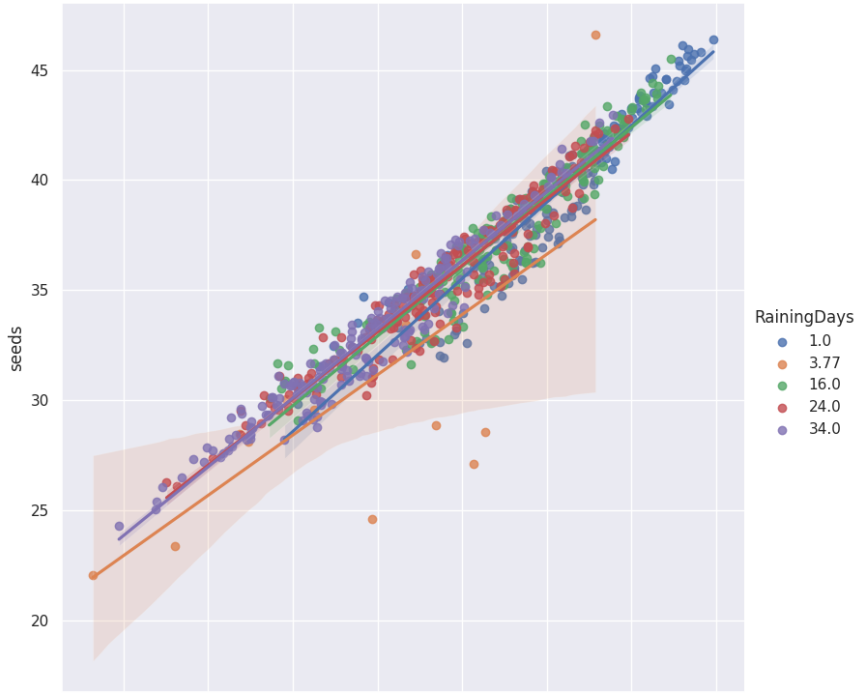

```
plt.subplots(figsize=(12,8))
sns.regplot(x="yield",y="seeds", data=df)
```

```
<Axes: xlabel='yield', ylabel='seeds'>
```



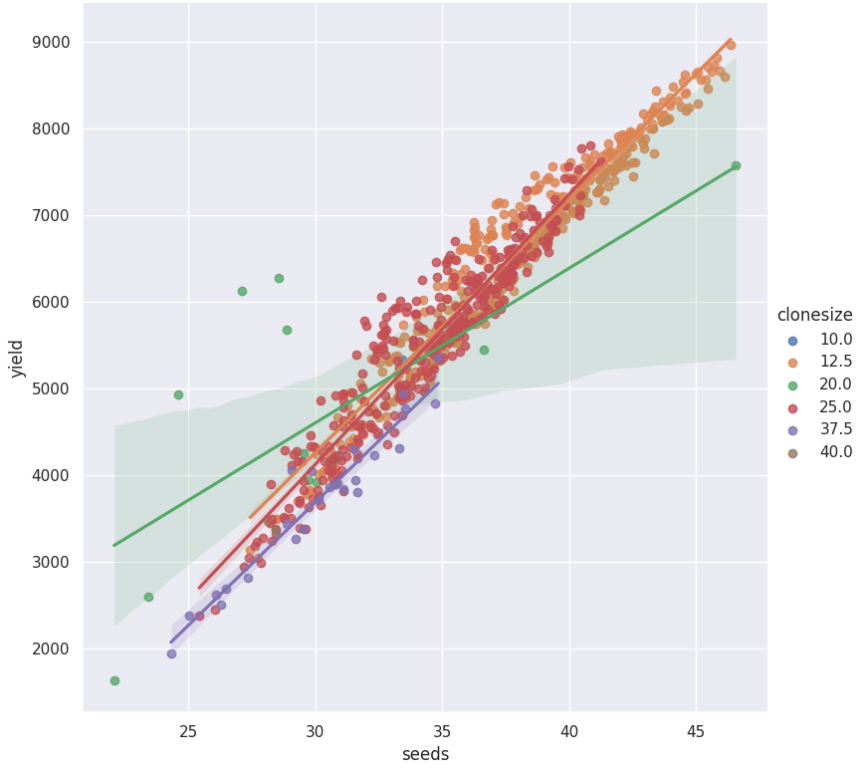

```
sns.lmplot(x="yield",y="seeds",data=df,hue="RainingDays",height=8)
```

```
<seaborn.axisgrid.FacetGrid at 0x7ab070f4b760>
```




```
sns.lmplot(x="seeds", y="yield", data=df, hue="clonesize",height=8)
```

```
<seaborn.axisgrid.FacetGrid at 0x7ab071326ef0>
```




```
yeild= df[df["yield"]>=0.8]
yeild.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 777 entries, 0 to 776
Data columns (total 11 columns):
 #   Column                Non-Null Count  Dtype
```

```
 ---  ------                --------------  -----
  0   Row#                  777 non-null    int64
  1   clonesize             777 non-null    float64
  2   MaxOfLowerTRange      777 non-null    float64
  3   MinOfLowerTRange      777 non-null    float64
  4   AverageOfLowerTRange  777 non-null    float64
  5   RainingDays           777 non-null    float64
  6   AverageRainingDays    777 non-null    float64
  7   fruitset              777 non-null    float64
  8   fruitmass             777 non-null    float64
  9   seeds                 777 non-null    float64
  10  yield                 777 non-null    float64
dtypes: float64(10), int64(1)
memory usage: 72.8 KB
```

```
yeild.corr()
```

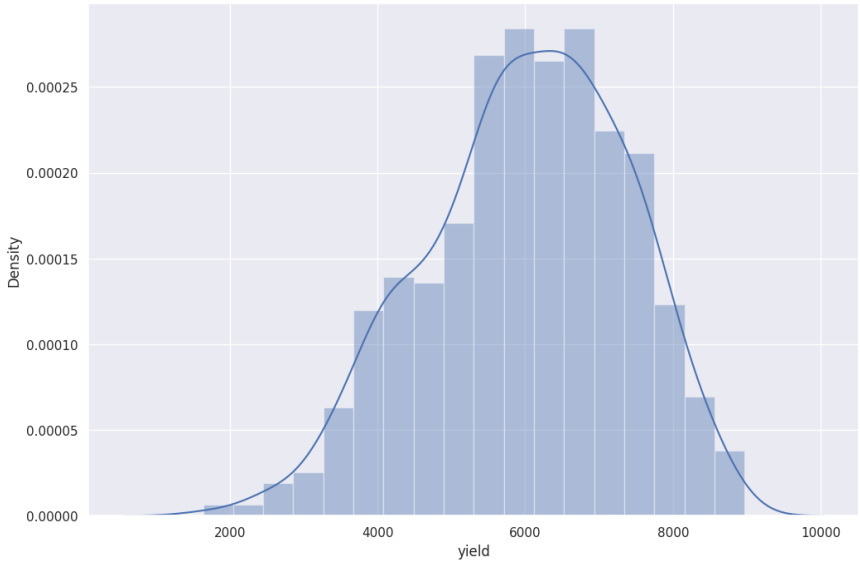
	Row#	clonesize	MaxOfLowerTRange	MinOfLowerTRange	A
Row#	1.000000	0.438706	-0.016947	-0.022430	
clonesize	0.438706	1.000000	0.034295	0.033768	
MaxOfLowerTRange	-0.016947	0.034295	1.000000	0.998071	
MinOfLowerTRange	-0.022430	0.033768	0.998071	1.000000	
AverageOfLowerTRange	-0.024338	0.033566	0.996609	0.999787	
RainingDays	-0.025894	-0.021696	-0.003558	-0.002403	
AverageRainingDays	-0.036481	-0.024455	-0.006087	-0.004334	
fruitset	-0.075130	-0.556591	-0.130693	-0.126788	
fruitmass	-0.077495	-0.474038	0.058487	0.062093	
seeds	-0.078344	-0.496156	-0.034674	-0.030727	
yield	-0.024942	-0.516737	-0.187439	-0.183339	

```
plt.subplots(figsize=(12,8))
sns.set_theme(style="darkgrid")
sns.distplot( yeild["seeds"])
```

```
<ipython-input-144-a5f457217d7a>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
plt.subplots(figsize=(12,8))
sns.set_theme(style="darkgrid")
sns.distplot( yeild["yield"])
```

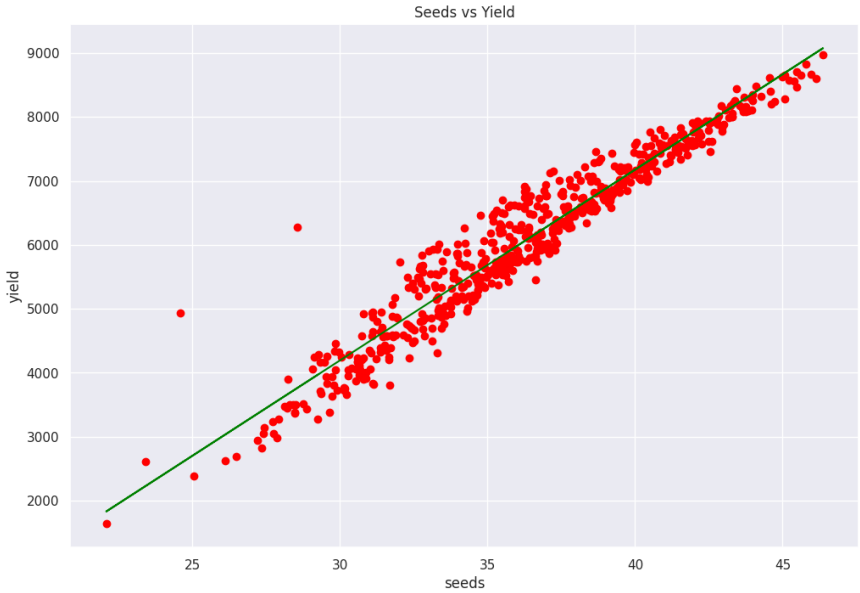
```
<ipython-input-145-eee47318cc1f>:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot( yeild["yield"])
<Axes: xlabel='yield', ylabel='Density'>
```




```
X=df["seeds"].values
y=df["yield"].values

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.25)

from sklearn.linear_model import LinearRegression
lr= LinearRegression()

lr.fit(X_train.reshape(-1,1), y_train)
y_pred = lr.predict(X_test.reshape(-1,1))

lr.score(X_test.reshape(-1,1),y_test.reshape(-1,1))
```

```
0.9029425040990746
```

```
plt.subplots(figsize=(12,8))
plt.scatter(X_train, y_train, color = "red")
plt.plot(X_train, lr.predict(X_train.reshape(-1,1)), color = "green")
plt.title("Seeds vs Yield")
plt.xlabel("seeds")
plt.ylabel("yield")
plt.show()
```




```
test= 320
val= test/340
val_out=lr.predict(np.array([[val]]))
print("yield", val_out[0])
```

```
yield -4477.556060085579
```

```
df.head()
```

	Row#	clonesize	MaxOfLowerTRange	MinOfLowerTRange	AverageOfLowerTRange	F
0	0	37.5	62.0	30.0	50.8	
1	1	37.5	62.0	30.0	50.8	
2	2	37.5	68.2	33.0	55.9	
3	3	37.5	68.2	33.0	55.9	
4	4	37.5	62.0	30.0	50.8	

```
df['MaxOfLowerTRange'] = [1 if each > 62 else 0 for each in df['MaxOfLowerTRange']]
df.head()
```

	Row#	clonesize	MaxOfLowerTRange	MinOfLowerTRange	AverageOfLowerTRange	F
0	0	37.5	0	30.0	50.8	
1	1	37.5	0	30.0	50.8	
2	2	37.5	1	33.0	55.9	
3	3	37.5	1	33.0	55.9	
4	4	37.5	0	30.0	50.8	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 777 entries, 0 to 776
Data columns (total 11 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Row#                 777 non-null    int64  
 1   clonesize            777 non-null    float64
 2   MaxOfLowerTRange     777 non-null    int64  
 3   MinOfLowerTRange     777 non-null    float64
 4   AverageOfLowerTRange  777 non-null    float64
 5   RainingDays          777 non-null    float64
 6   AverageRainingDays   777 non-null    float64
 7   fruitset             777 non-null    float64
 8   fruitmass            777 non-null    float64
 9   seeds                777 non-null    float64
 10  yield                777 non-null    float64
dtypes: float64(9), int64(2)
memory usage: 66.9 KB
```

```
x = df[['MaxOfLowerTRange','seeds']]
```

```
df['seeds' ] = [1 if each > 30 else 0 for each in df['yield']]
```

```
y = df['yield']
```

```
df.head()
df['yield' ] = [1 if each > 4000 else 0 for each in df['yield']]
df.head()
```

	Row#	clonesize	MaxOfLowerTRange	MinOfLowerTRange	AverageOfLowerTRange	R
0	0	37.5	62.0	30.0	50.8	
1	1	37.5	62.0	30.0	50.8	
2	2	37.5	68.2	33.0	55.9	
3	3	37.5	68.2	33.0	55.9	
4	4	37.5	62.0	30.0	50.8	

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.25,random_state=1)
```

```
print(f"Size of splitted data")
print(f"x_train {x_train.shape}")
print(f"y_train {y_train.shape}")
print(f"y_train {x_test.shape}")
print(f"y_test {y_test.shape}")
```

```
Size of splitted data
x_train (582, 2)
y_train (582,)
y_train (195, 2)
y_test (195,)
```

```
from  sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LogisticRegression
model_dt = DecisionTreeRegressor(random_state=1)
model_rf = RandomForestRegressor(random_state=1)
model_lr = LogisticRegression(random_state=1,solver='lbfgs',max_iter=1000)
```

```
model_dt.fit(x_train,y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor(random_state=1)
```

```
model_rf.fit(x_train,y_train)
```

```
RandomForestRegressor
```

```
model_lr.fit(x_train,y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000, random_state=1)
```

```
y_pred_dt = model_dt.predict(x_test) #int
y_pred_rf = model_rf.predict(x_test) #float
y_pred_lr = model_lr.predict(x_test) #
```

```
result= pd.DataFrame({ "Actual": y_test, "predicted": y_pred_dt})
result
```

	Actual	predicted
374	1	1.0
491	1	1.0
678	1	1.0
720	1	1.0
412	1	1.0
...	...	...
139	1	1.0
218	0	0.0
467	1	1.0
118	1	1.0
201	1	1.0

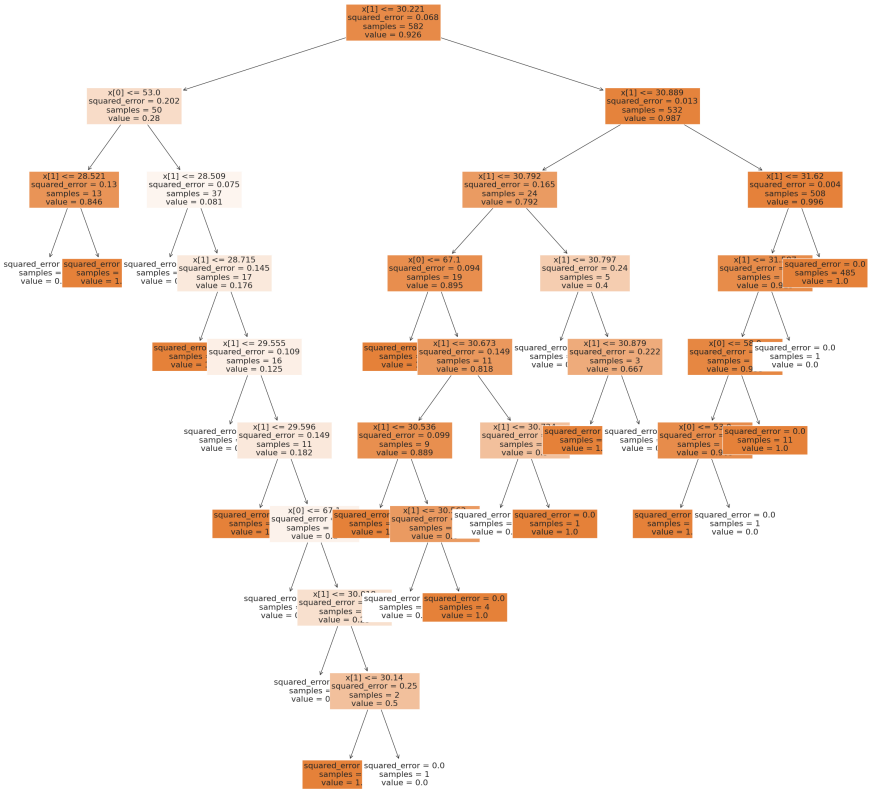
195 rows × 2 columns

```
y_pred_rf = [1 if each > 0.75 else 0 for each in y_pred_rf]
```

```
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score
from sklearn.metrics import classification_report
```

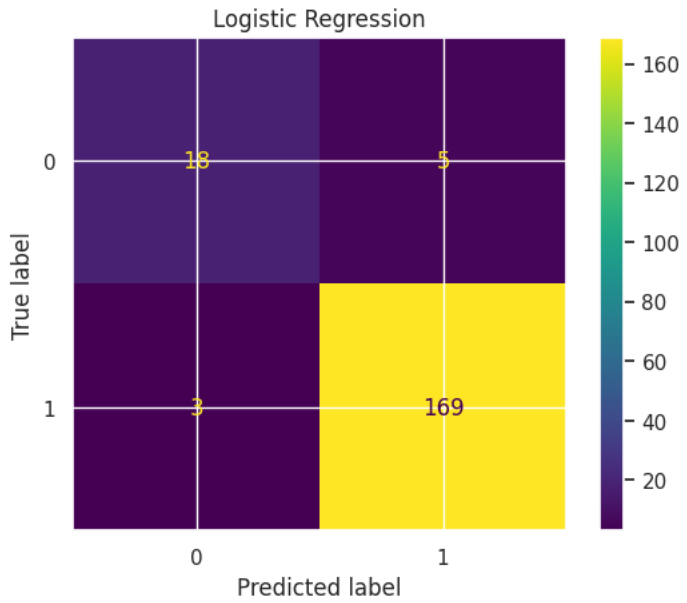
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_dt)
plt.title('Decision Tree')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_dt)}")
print(classification_report(y_test,y_pred_dt))
```


```
from sklearn import tree
import matplotlib.pyplot as plt
plt.figure(figsize=(30,30))
tree.plot_tree(model_dt, filled=True, fontsize=16)
plt.show()
```



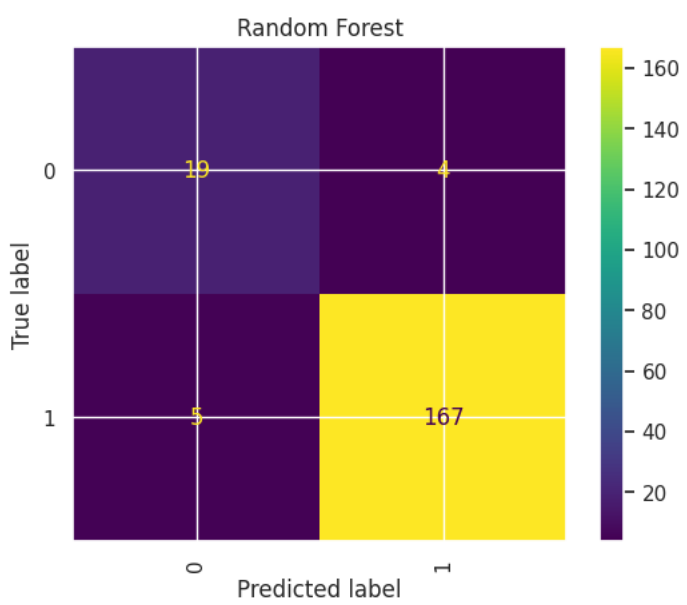

```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_lr)
plt.title('Logistic Regression')
```

```
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_lr)}")
print(classification_report(y_test,y_pred_lr))
```




```
Accuracy is 0.958974358974359
              precision    recall  f1-score   support

           0       0.86      0.78      0.82        23
           1       0.97      0.98      0.98       172

    accuracy                           0.96       195
   macro avg       0.91      0.88      0.90       195
weighted avg       0.96      0.96      0.96       195
```

```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_rf,xticks_rotation='vertical')
plt.title('Random Forest')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_rf)}")
print(classification_report(y_test,y_pred_rf))
```




```
Accuracy is 0.9538461538461539
              precision    recall  f1-score   support

           0       0.79      0.83      0.81        23
           1       0.98      0.97      0.97       172

    accuracy                           0.95       195
   macro avg       0.88      0.90      0.89       195
weighted avg       0.95      0.95      0.95       195
```