Crop Yield Recommendation

In [1]:

*#Importing Libraries* **import** pandas **as** pd **import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** warnings warnings.filterwarnings('ignore')

In [2]:

*#Reading the dataset*

crop**=**pd.read\_csv("C:\\Users\\GPT BANTWAL\\Downloads\\archive (4)\\Crop\_r crop

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[2]: |  | | | | | | | | |
|  |  | **Nitrogen** | **phosphorus** | **potassium** | **temperature** | **humidity** | **ph** | **rainfall** | **lab** |
|  | **0** | 90 | 42 | 43 | 20.879744 | 82.002744 | 6.502985 | 202.935536 | ric |
|  | **1** | 85 | 58 | 41 | 21.770462 | 80.319644 | 7.038096 | 226.655537 | ric |
|  | **2** | 60 | 55 | 44 | 23.004459 | 82.320763 | 7.840207 | 263.964248 | ric |
|  | **3** | 74 | 35 | 40 | 26.491096 | 80.158363 | 6.980401 | 242.864034 | ric |
|  | **4** | 78 | 42 | 42 | 20.130175 | 81.604873 | 7.628473 | 262.717340 | ric |
|  | **...** | ... | ... | ... | ... | ... | ... | ... |  |
|  | **2195** | 107 | 34 | 32 | 26.774637 | 66.413269 | 6.780064 | 177.774507 | coffe |
|  | **2196** | 99 | 15 | 27 | 27.417112 | 56.636362 | 6.086922 | 127.924610 | coffe |
|  | **2197** | 118 | 33 | 30 | 24.131797 | 67.225123 | 6.362608 | 173.322839 | coffe |
|  | **2198** | 117 | 32 | 34 | 26.272418 | 52.127394 | 6.758793 | 127.175293 | coffe |
|  | **2199** | 104 | 18 | 30 | 23.603016 | 60.396475 | 6.779833 | 140.937041 | coffe |

2200 rows × 10 columns

In [3]:

crop.drop(['Unnamed: 8','Unnamed: 9'],axis**=**1,inplace**=True**) crop

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[3]: |  | | | | | | | | |
|  |  | **Nitrogen** | **phosphorus** | **potassium** | **temperature** | **humidity** | **ph** | **rainfall** | **lab** |
|  | **0** | 90 | 42 | 43 | 20.879744 | 82.002744 | 6.502985 | 202.935536 | ric |
|  | **1** | 85 | 58 | 41 | 21.770462 | 80.319644 | 7.038096 | 226.655537 | ric |
|  | **2** | 60 | 55 | 44 | 23.004459 | 82.320763 | 7.840207 | 263.964248 | ric |
|  | **3** | 74 | 35 | 40 | 26.491096 | 80.158363 | 6.980401 | 242.864034 | ric |
|  | **4** | 78 | 42 | 42 | 20.130175 | 81.604873 | 7.628473 | 262.717340 | ric |
|  | **...** | ... | ... | ... | ... | ... | ... | ... |  |
|  | **2195** | 107 | 34 | 32 | 26.774637 | 66.413269 | 6.780064 | 177.774507 | coffe |
|  | **2196** | 99 | 15 | 27 | 27.417112 | 56.636362 | 6.086922 | 127.924610 | coffe |
|  | **2197** | 118 | 33 | 30 | 24.131797 | 67.225123 | 6.362608 | 173.322839 | coffe |
|  | **2198** | 117 | 32 | 34 | 26.272418 | 52.127394 | 6.758793 | 127.175293 | coffe |
|  | **2199** | 104 | 18 | 30 | 23.603016 | 60.396475 | 6.779833 | 140.937041 | coffe |

2200 rows × 8 columns

In [4]:

crop.head()

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[4]: |  | | | | | | | | |
|  |  | **Nitrogen** | **phosphorus** | **potassium** | **temperature** | **humidity** | **ph** | **rainfall** | **label** |
|  | **0** | 90 | 42 | 43 | 20.879744 | 82.002744 | 6.502985 | 202.935536 | rice |
|  | **1** | 85 | 58 | 41 | 21.770462 | 80.319644 | 7.038096 | 226.655537 | rice |
|  | **2** | 60 | 55 | 44 | 23.004459 | 82.320763 | 7.840207 | 263.964248 | rice |
|  | **3** | 74 | 35 | 40 | 26.491096 | 80.158363 | 6.980401 | 242.864034 | rice |
|  | **4** | 78 | 42 | 42 | 20.130175 | 81.604873 | 7.628473 | 262.717340 | rice |

In [5]:

crop.tail()

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[5]: |  | | | | | | | | |
|  |  | **Nitrogen** | **phosphorus** | **potassium** | **temperature** | **humidity** | **ph** | **rainfall** | **lab** |
|  | **2195** | 107 | 34 | 32 | 26.774637 | 66.413269 | 6.780064 | 177.774507 | coffe |
|  | **2196** | 99 | 15 | 27 | 27.417112 | 56.636362 | 6.086922 | 127.924610 | coffe |
|  | **2197** | 118 | 33 | 30 | 24.131797 | 67.225123 | 6.362608 | 173.322839 | coffe |
|  | **2198** | 117 | 32 | 34 | 26.272418 | 52.127394 | 6.758793 | 127.175293 | coffe |
|  | **2199** | 104 | 18 | 30 | 23.603016 | 60.396475 | 6.779833 | 140.937041 | coffe |

In [6]:

crop.isna().sum()

Out[6]: Nitrogen 0

phosphorus 0

potassium 0

temperature 0

humidity 0

ph 0

rainfall 0

label 0

dtype: int64

In [7]:

print("The size of the dataset:",crop.size) print("Shape of the dataset:",crop.shape)

The size of the dataset: 17600 Shape of the dataset: (2200, 8)

In [8]:

print("Number of columns in the dataset:\n",crop.columns)

Number of columns in the dataset:

Index(['Nitrogen', 'phosphorus', 'potassium', 'temperature', 'humidit y', 'ph',

'rainfall', 'label'], dtype='object')

In [9]:

print("The unique labels in the datasets are:",crop['label'].unique())

The unique labels in the datasets are: ['rice' 'maize' 'chickpea' 'kidn eybeans' 'pigeonpeas' 'mothbeans'

'mungbean' 'blackgram' 'lentil' 'pomegranate' 'banana' 'mango' 'grape s'

'watermelon' 'muskmelon' 'apple' 'orange' 'papaya' 'coconut' 'cotton' 'jute' 'coffee']

In [10]:

print("Value count of the labels:\n",crop['label'].value\_counts())

Value count of the labels: label

|  |  |  |
| --- | --- | --- |
| rice | 100 |  |
| maize | 100 |  |
| jute | 100 |  |
| cotton | 100 |  |
| coconut | 100 |  |
| papaya | 100 |  |
| orange | 100 |  |
| apple | 100 |  |
| muskmelon | 100 |  |
| watermelon | 100 |  |
| grapes | 100 |  |
| mango | 100 |  |
| banana | 100 |  |
| pomegranate | 100 |  |
| lentil | 100 |  |
| blackgram | 100 |  |
| mungbean | 100 |  |
| mothbeans | 100 |  |
| pigeonpeas | 100 |  |
| kidneybeans | 100 |  |
| chickpea | 100 |  |
| coffee | 100 |  |
| Name: count, | dtype: | int64 |

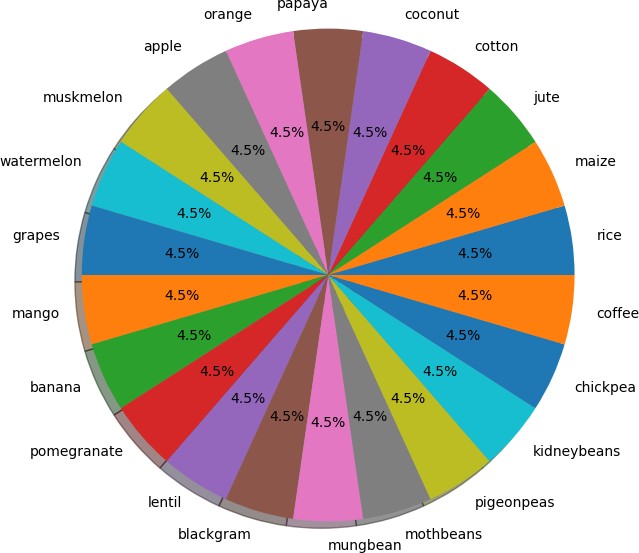
**Data visualization**

In [11]:

*#Pie chart for label column* data\_viz\_df **=** crop.copy() data\_viz\_df.head()

label\_name **=** data\_viz\_df['label'].value\_counts().index val **=** data\_viz\_df['label'].value\_counts().values plt.figure(figsize **=** (8,8))

plt.pie(x **=** val , labels **=** label\_name , shadow **= True** , autopct **=** '%1.1 plt.show()

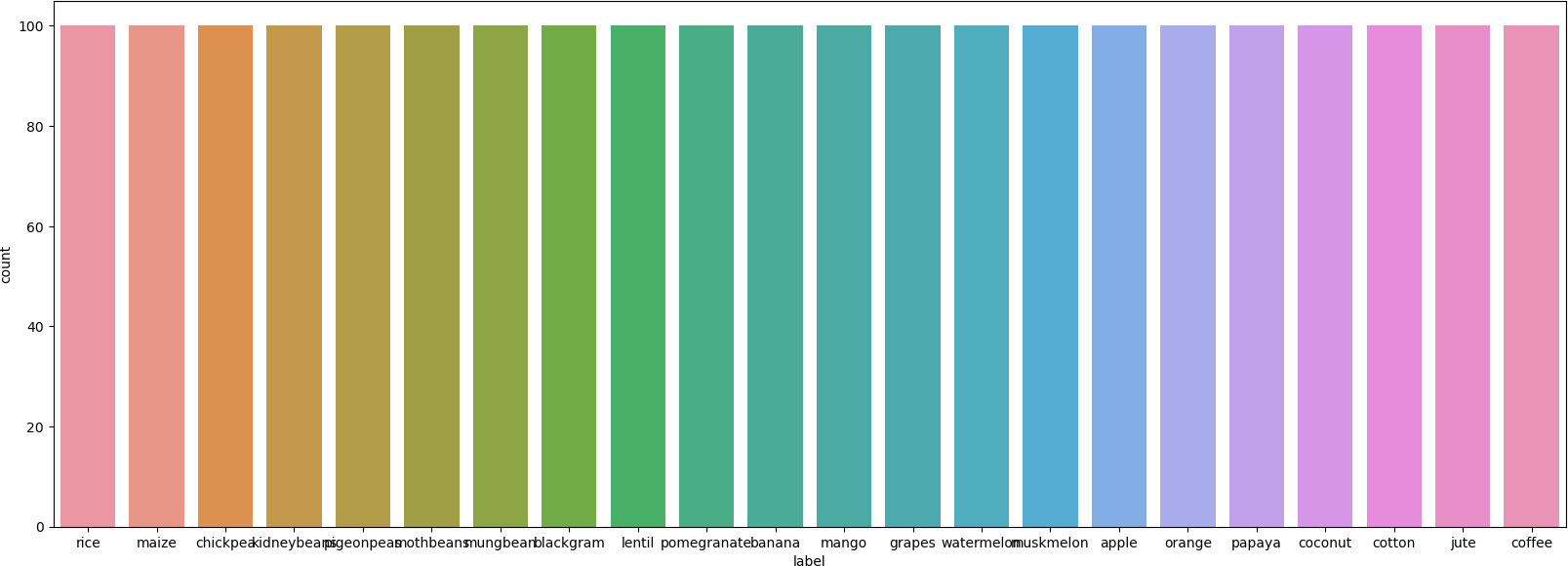


In [12]:

*#Bar graph for Label column*

plt.figure(figsize **=** (20 , 7))

sns.countplot(x **=** 'label' , data **=** data\_viz\_df) plt.show()



In [13]:

*#Label Encoding*

**from** sklearn.preprocessing **import** LabelEncoder le **=** LabelEncoder()

crop['label'] **=** le.fit\_transform(crop['label']) crop.head()

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[13]: |  | | | | | | | | |
|  |  | **Nitrogen** | **phosphorus** | **potassium** | **temperature** | **humidity** | **ph** | **rainfall** | **label** |
|  | **0** | 90 | 42 | 43 | 20.879744 | 82.002744 | 6.502985 | 202.935536 | 20 |
|  | **1** | 85 | 58 | 41 | 21.770462 | 80.319644 | 7.038096 | 226.655537 | 20 |
|  | **2** | 60 | 55 | 44 | 23.004459 | 82.320763 | 7.840207 | 263.964248 | 20 |
|  | **3** | 74 | 35 | 40 | 26.491096 | 80.158363 | 6.980401 | 242.864034 | 20 |
|  | **4** | 78 | 42 | 42 | 20.130175 | 81.604873 | 7.628473 | 262.717340 | 20 |

In [14]:

x**=**crop.drop('label',axis**=**1) x

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[14]: |  | | | | | | | |
|  |  | **Nitrogen** | **phosphorus** | **potassium** | **temperature** | **humidity** | **ph** | **rainfall** |
|  | **0** | 90 | 42 | 43 | 20.879744 | 82.002744 | 6.502985 | 202.935536 |
|  | **1** | 85 | 58 | 41 | 21.770462 | 80.319644 | 7.038096 | 226.655537 |
|  | **2** | 60 | 55 | 44 | 23.004459 | 82.320763 | 7.840207 | 263.964248 |
|  | **3** | 74 | 35 | 40 | 26.491096 | 80.158363 | 6.980401 | 242.864034 |
|  | **4** | 78 | 42 | 42 | 20.130175 | 81.604873 | 7.628473 | 262.717340 |
|  | **...** | ... | ... | ... | ... | ... | ... | ... |
|  | **2195** | 107 | 34 | 32 | 26.774637 | 66.413269 | 6.780064 | 177.774507 |
|  | **2196** | 99 | 15 | 27 | 27.417112 | 56.636362 | 6.086922 | 127.924610 |
|  | **2197** | 118 | 33 | 30 | 24.131797 | 67.225123 | 6.362608 | 173.322839 |
|  | **2198** | 117 | 32 | 34 | 26.272418 | 52.127394 | 6.758793 | 127.175293 |
|  | **2199** | 104 | 18 | 30 | 23.603016 | 60.396475 | 6.779833 | 140.937041 |

2200 rows × 7 columns

In [15]:

y**=**crop['label'] y

|  |  |  |  |
| --- | --- | --- | --- |
| Out[15]: | 0 | 20 |  |
|  | 1 | 20 |  |
|  | 2 | 20 |  |
|  | 3 | 20 |  |
|  | 4 | 20 |  |
|  |  | .. |  |
|  | 2195 | 5 |  |
|  | 2196 | 5 |  |
|  | 2197 | 5 |  |
|  | 2198 | 5 |  |
|  | 2199 | 5 |  |
|  | Name: | label, Length: 2200, | dtype: int32 |

In [16]:

*#Train\_Test\_Split*

**from** sklearn.model\_selection **import** train\_test\_split xtrain,xtest,ytrain,ytest**=**train\_test\_split(x,y,test\_size**=**0.33,random\_sta print("xtrain shape:",xtrain.shape)

print("xtest shape:",xtest.shape) print("ytrain shape:",ytrain.shape) print("ytest shape:",ytest.shape)

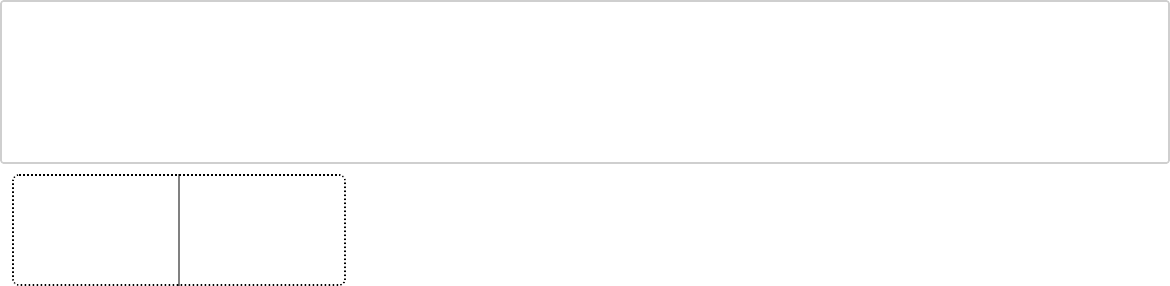
xtrain shape: (1474, 7)

xtest shape: (726, 7)

ytrain shape: (1474,)

ytest shape: (726,)

In [17]:



*#LogisticRegression()*

**from** sklearn.linear\_model **import** LogisticRegression LG **=** LogisticRegression()

LG.fit(xtrain,ytrain)

▾ LogisticRegression LogisticRegression()

Out[17]:

In [18]:

*#To check intercept and co-efficient* print('Intercept:',LG.intercept\_) print('Co-efficient:',LG.coef\_)

Intercept: [-0.00780263 -0.01053188 -0.01657646 0.00332404 -0.00463209

0.00217818

-0.01441516 -0.00392094 -0.00192072 0.01139565 0.00604579 0.0198111

9

0.02530412 0.04437448 -0.00247395 0.00067193 0.00319134 -0.0408240

7

-0.00357377 0.00243646 -0.0123666 0.00030511]

Co-efficient: [[-4.95073539e-01 2.37818463e-01 8.81830781e-01 -2.0263 2782e-01

-2.15887893e-01 -5.59788261e-02 -2.65260052e-01]

[ 4.05127196e-01 1.36164343e-01 1.71382861e-01 -1.54910028e-01

-2.87219896e-01 -8.73894751e-02 6.77970652e-02]

[-1.22341171e-01 4.17528456e-01 -4.32713716e-01 1.84071580e-01 6.03238933e-02 -6.94249955e-02 6.43785051e-02]

[ 7.08325385e-02 2.80986927e-01 4.72444155e-01 -2.58743675e-03

-4.93916314e-01 3.47613308e-02 7.42472165e-02]

[-3.39646106e-01 -5.62047249e-01 -1.45093905e-01 -4.47995485e-02 2.49809507e-01 -5.68238530e-02 3.57348946e-01]

[ 6.51856451e-01 -4.63425749e-01 -1.74397933e-01 7.00591314e-02

-4.16995828e-01 2.76566216e-02 2.26341219e-01]

[ 6.07064249e-01 -1.34120658e-01 -5.74851120e-01 -1.85494961e-01 6.30183486e-02 -2.64961112e-02 -1.90338661e-02]

[-4.88437717e-01 3.87346470e-01 1.04059788e+00 -1.27823008e-01

-2.31305124e-01 -2.88194637e-02 -8.50592522e-01]

[ 1.87619165e-01 -1.72504515e-01 8.65554733e-03 -1.27720111e-01

-8.28985985e-02 6.62275793e-02 2.66109092e-01]

[-1.94869901e-01 6.75571637e-01 -6.24381988e-02 2.12318371e-02

-7.24283312e-01 6.29970896e-02 1.97017402e-01]

[-2.03451136e-01 5.39442364e-01 -3.12328316e-01 -1.59233240e-01 2.65899974e-01 5.12321949e-02 -1.67062599e-01]

[ 4.51779083e-01 -4.11378491e-02 -3.08784165e-01 5.16555678e-02

-7.66516964e-02 4.94237098e-02 9.55320543e-02]

[-2.92335847e-01 -1.84513824e-01 3.09791520e-01 5.71415473e-01

-8.67931914e-02 9.61871797e-02 1.93947218e-01]

[-2.00695130e-01 1.74360074e-01 -7.08670813e-02 8.14513378e-01

-2.31984400e-02 3.17598216e-01 -5.12919712e-02]

[-2.48552052e-01 2.41566568e-02 -4.41589527e-01 4.31564688e-03 5.82246903e-01 -1.62198260e-02 -7.35054279e-02]

[ 5.58967586e-01 -3.63881002e-01 1.26054026e-01 7.76629528e-02 3.39146030e-01 1.00747331e-03 -8.97769017e-01]

[-3.13038606e-01 -3.93235455e-01 -5.19932493e-01 -1.69364685e-02 5.04927164e-01 4.25652674e-02 1.96768524e-01]

[-2.54664317e-01 1.16082950e-01 4.07403182e-01 -2.03504754e-01 1.11101461e-01 -2.34937402e-01 1.65238198e-01]

[-2.47777790e-01 3.75688234e-01 -5.17238753e-01 8.52964352e-02

-4.56962203e-02 -5.08523674e-02 2.60099466e-01]

[-3.01581802e-01 -3.96251414e-01 2.55926826e-01 -1.43333184e-01 4.29057811e-01 2.21824647e-02 1.02037954e-01]

[ 1.91757191e-01 -2.23053767e-01 -2.53887667e-01 -4.75648095e-01

-2.32196394e-02 -1.46840769e-01 3.54096352e-01]

[ 5.77461657e-01 -4.30975091e-01 1.40036099e-01 -3.55983858e-02 1.02535061e-01 1.94396179e-03 -2.96443758e-01]]

In [19]:

*#Predictions* y\_pred**=**LG.predict(xtest) y\_pred

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Out[19]: | array([21, | 21, | 7, | 3, | 2, | 8, | 13, | 9, | 15, | 1, | 13, | 5, | 10, | 14, | 12, | 0, |
|  | 5, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 10, | 5, | 12, | 4, | 2, | 9, | 8, | 6, | 5, | 10, | 16, | 13, | 9, | 19, | 20, | 11, |
|  | 15, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 4, | 6, | 12, | 12, | 21, | 13, | 11, | 2, | 18, | 21, | 18, | 14, | 9, | 9, | 6, | 14, |
|  | 13, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 2, | 0, | 15, | 18, | 1, | 17, | 12, | 10, | 6, | 16, | 14, | 21, | 20, | 15, | 0, | 7, |
|  | 5, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 0, | 16, | 4, | 19, | 9, | 11, | 7, | 13, | 3, | 11, | 8, | 12, | 20, | 13, | 21, | 21, |
|  | 15, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 6, | 11, | 10, | 13, | 17, | 2, | 8, | 14, | 7, | 14, | 11, | 5, | 8, | 10, | 3, | 16, |
|  | 8, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 14, | 1, | 1, | 20, | 21, | 5, | 18, | 15, | 15, | 12, | 5, | 7, | 16, | 19, | 14, | 10, |
|  | 11, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 8, | 19, | 10, | 16, | 3, | 3, | 2, | 19, | 16, | 3, | 12, | 13, | 2, | 15, | 14, | 6, |
|  | 14, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 4, | 19, | 16, | 2, | 10, | 7, | 0, | 5, | 3, | 0, | 8, | 12, | 21, | 17, | 16, | 4, |
|  | 13, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1, | 19, | 3, | 21, | 11, | 0, | 8, | 10, | 18, | 8, | 9, | 9, | 15, | 20, | 15, | 1, |
|  | 16, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 9, | 0, | 13, | 4, | 6, | 14, | 9, | 19, | 17, | 16, | 20, | 17, | 17, | 9, | 9, | 1, |
|  | 4, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 18, | 20, | 17, | 11, | 8, | 13, | 20, | 11, | 5, | 18, | 4, | 3, | 12, | 4, | 19, | 6, |
|  | 13, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 18, | 16, | 15, | 11, | 18, | 1, | 3, | 2, | 18, | 16, | 13, | 14, | 12, | 17, | 15, | 19, |
|  | 8, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 20, | 2, | 17, | 2, | 5, | 11, | 5, | 16, | 20, | 13, | 14, | 16, | 9, | 19, | 4, | 12, |
|  | 14, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 6, | 20, | 3, | 14, | 0, | 18, | 13, | 20, | 21, | 2, | 19, | 16, | 11, | 7, | 3, | 18, |
|  | 8, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 17, | 19, | 5, | 12, | 13, | 8, | 21, | 19, | 20, | 7, | 4, | 8, | 10, | 3, | 5, | 5, |
|  | 17, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 19, | 11, | 20, | 3, | 18, | 16, | 19, | 18, | 4, | 9, | 19, | 15, | 13, | 12, | 10, | 1, |
|  | 2, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 12, | 9, | 12, | 6, | 14, | 4, | 7, | 7, | 18, | 17, | 20, | 20, | 3, | 15, | 5, | 21, |
|  | 8, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 8, | 13, | 7, | 15, | 2, | 13, | 13, | 3, | 2, | 12, | 1, | 12, | 19, | 8, | 16, | 15, |
|  | 3, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 10, | 6, | 17, | 7, | 9, | 10, | 0, | 20, | 15, | 0, | 17, | 2, | 8, | 3, | 13, | 10, |
|  | 7, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 20, | 9, | 15, | 12, | 7, | 17, | 20, | 5, | 15, | 13, | 1, | 17, | 16, | 9, | 21, | 18, |
|  | 0, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 21, | 21, | 18, | 9, | 2, | 9, | 8, | 4, | 6, | 9, | 16, | 6, | 18, | 19, | 6, | 6, |
|  | 0, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 6, | 0, | 16, | 11, | 7, | 1, | 0, | 13, | 20, | 9, | 1, | 20, | 10, | 3, | 19, | 1, |
|  | 3, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 15, | 19, | 0, | 10, | 15, | 16, | 2, | 15, | 13, | 12, | 3, | 19, | 12, | 3, | 4, | 15, |
|  | 1, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 18, | 17, | 8, | 2, | 6, | 20, | 1, | 4, | 20, | 2, | 11, | 16, | 21, | 20, | 0, | 7, |
|  | 18, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 7, | 3, | 12, | 8, | 19, | 11, | 12, | 7, | 1, | 14, | 18, | 1, | 6, | 2, | 0, | 0, |
|  | 8, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 8, | 21, | 3, | 1, | 19, | 1, | 9, | 7, | 11, | 5, | 6, | 8, | 7, | 5, | 14, | 2, |
|  | 8, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 16, | 18, | 18, | 15, | 2, | 21, | 14, | 21, | 17, | 14, | 14, | 14, | 19, | 16, | 13, | 0, |
|  | 5, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 4, | 11, | 4, | 7, | 7, | 3, | 3, | 12, | 9, | 12, | 16, | 14, | 17, | 18, | 2, | 17, |
|  | 15, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 2, | 1, | 20, | 5, | 6, | 7, | 8, | 3, | 15, | 1, | 7, | 21, | 15, | 9, | 8, | 18, |
|  | 6, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 21, | 19, | 5, | 4, | 11, | 20, | 14, | 9, | 21, | 14, | 0, | 0, | 21, | 1, | 18, | 14, |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0, |  | | | | | | | | | | | | | | | |
|  | 14, | 6, | 20, | 17, | 6, | 17, | 3, | 0, | 19, | 13, | 20, | 2, | 12, | 16, | 8, | 1, |
| 13, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 5, | 6, | 12, | 5, | 4, | 19, | 6, | 7, | 2, | 3, | 8, | 3, | 17, | 16, | 6, | 1, |
| 2, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 15, | 17, | 0, | 16, | 19, | 11, | 18, | 17, | 12, | 19, | 17, | 7, | 20, | 6, | 8, | 13, |
| 10, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 13, | 9, | 1, | 13, | 0, | 17, | 21, | 4, | 3, | 10, | 9, | 13, | 7, | 7, | 16, | 20, |
| 2, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1, | 6, | 6, | 13, | 20, | 20, | 4, | 13, | 6, | 5, | 17, | 5, | 14, | 10, | 16, | 19, |
| 3, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 10, | 6, | 12, | 16, | 5, | 20, | 17, | 17, | 4, | 20, | 6, | 13, | 4, | 20, | 7, | 0, |
| 1, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 4, | 1, | 11, | 12, | 17, | 17, | 20, | 8, | 15, | 6, | 10, | 9, | 2, | 5, | 20, | 16, |
| 4, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1, | 2, | 0, | 11, | 12, | 3, | 4, | 15, | 5, | 19, | 16, | 7, | 17, | 3, | 8, | 21, |
| 16, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 9, | 16, | 16, | 10, | 11, | 12, | 9, | 19, | 4, | 13, | 11, | 10, | 14, | 20, | 9, | 16, |
| 10, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 5, | 14, | 15, | 4, | 7, | 4, | 19, | 18, | 4, | 10, | 17, | 1, | 3, | 13, | 17, | 16, |
| 10, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 19, | 2, | 20, | 16, | 20, | 3, | 2, | 18, | 5, | 3, | 7, | 4, | 3, | 7, | 5, | 19, |
| 19, |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1, | 3, | 2, | 18, | 13, | 0, | 19, | 0, | 13, | 0, | 21, | 18]) |  |  |  |  |

In [20]:

*#Evaluating the model*

**from** sklearn.metrics **import** mean\_squared\_error,mean\_absolute\_error train\_accuracy**=**LG.score(xtrain,ytrain) print('Train\_accuracy(R\_Squared):',train\_accuracy) test\_accuracy**=**LG.score(xtest,ytest) print('Test\_accuracy(R\_Squared):',test\_accuracy)

Train\_accuracy(R\_Squared): 0.9721845318860244

Test\_accuracy(R\_Squared): 0.9435261707988981

In [21]:

**import** math

**from** sklearn.metrics **import** mean\_squared\_error,mean\_absolute\_error print('Mean Absolute Error:',mean\_absolute\_error(ytest,y\_pred)) print('Mean Squared Error:',mean\_squared\_error(ytest,y\_pred)) print('Root Mean Squared Error:',math.sqrt(mean\_squared\_error(ytest,y\_pr

Mean Absolute Error: 0.47107438016528924 Mean Squared Error: 4.597796143250688

Root Mean Squared Error: 2.1442472206466046

In [22]:

*#To check the actual label,predicted label and its difference* dfr**=**pd.DataFrame({'Actual label':ytest,'Predicted label':y\_pred}) dfr

|  |  |  |  |
| --- | --- | --- | --- |
| Out[22]: |  | | |
|  |  | **Actual label** | **Predicted label** |
|  | **1320** | 21 | 21 |
|  | **1367** | 21 | 21 |
|  | **1291** | 7 | 7 |
|  | **264** | 3 | 3 |
|  | **728** | 2 | 2 |
|  | **...** | ... | ... |
|  | **1523** | 0 | 0 |
|  | **731** | 2 | 13 |
|  | **1545** | 0 | 0 |
|  | **1358** | 21 | 21 |
|  | **383** | 9 | 18 |

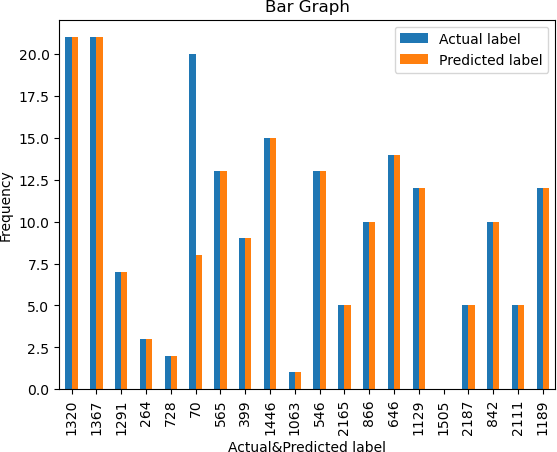
726 rows × 2 columns

In [23]:

*#Plotting the Bar graph for above actual,predicted label and difference*

graph**=**dfr.head(20) graph.plot(kind**=**'bar') plt.title('Bar Graph') plt.xlabel('Actual&Predicted label') plt.ylabel('Frequency')

plt.show()



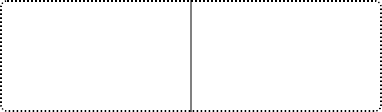
In [24]:

*#RandomForestClassifier()*

**from** sklearn.ensemble **import** RandomForestRegressor RF **=** RandomForestRegressor()

RF.fit(xtrain,ytrain)

Out[24]:



▾ RandomForestRegressor

RandomForestRegressor()

In [25]:

*#Evaluating model*

train\_accuracy **=** RF.score(xtrain,ytrain) print("Train\_accuracy(R\_Squared):",train\_accuracy) test\_accuracy **=** RF.score(xtest,ytest) print("Test\_accuracy(R\_Squared):",test\_accuracy)

Train\_accuracy(R\_Squared): 0.9956159882585227

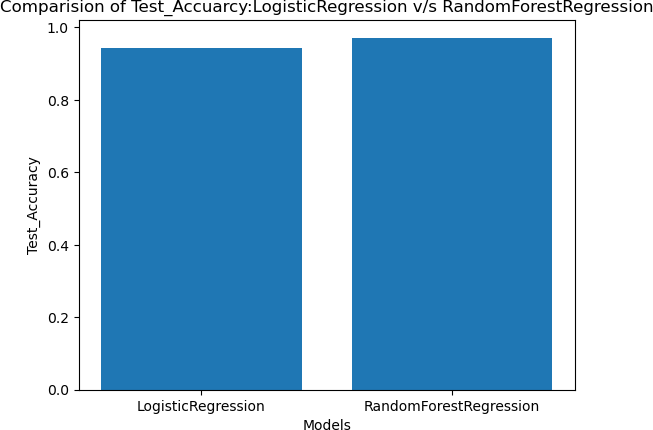
Test\_accuracy(R\_Squared): 0.9734312241479864

In [26]:

*#Comparision between Linear and RandomForestRegression using boxplot* logistic\_regression\_accuracy**=**0.9435261707988981 random\_forest\_accuracy**=**0.9715513770838728 accuracy\_scores**=**[logistic\_regression\_accuracy,random\_forest\_accuracy] model\_names**=**['LogisticRegression','RandomForestRegression'] plt.bar(model\_names,accuracy\_scores)

plt.xlabel('Models') plt.ylabel('Test\_Accuracy')

plt.title('Comparision of Test\_Accuarcy:LogisticRegression v/s RandomFor plt.show()



In [ ]: