|  |  |
| --- | --- |
| CODIGO MINIMOS CUADRADOS | |
| 1. import numpy as np 2. import matplotlib.pyplot as plt 3. #vectores para valor x 4. xi=np.array([1.0, 1.6, 3.4, 4.0, 5.2]) 5. #vectores para valor y 6. yi=np.array([1.2, 2.0, 2.4, 3.5, 3.5]) 7. #tamaño del vector 8. n=len(xi) 9. #operaciones de sumatorias y nombre de variables 10. Sum\_xi=sum(xi) 11. Sum\_yi=sum(yi) 12. Sum\_xx=sum(xi\*\*2) 13. Sum\_xy=sum(xi\*yi) 14. a=((n)\*(Sum\_xy)-(Sum\_xi)\*(Sum\_yi))/((n)\*(Sum\_xx)-(Sum\_xi)\*\*2) 15. b=((Sum\_yi)\*(Sum\_xx)-(Sum\_xy)\*(Sum\_xi))/((n)\*(Sum\_xx)-(Sum\_xi)\*\*2) 16. #impresion de resultados 17. print(Sum\_xi, Sum\_yi, Sum\_xx, Sum\_xy, a, b) 18. x=np.linspace(0, 5, 4) 19. y=a+b\*x 20. #Grafico de los puntos y la recta de ajuste-------------------------- 21. plt.figure(1) 22. plt.scatter(xi, yi, color='b') 23. plt.grid(linestyle='dotted') 24. plt.plot(x, y, color='g') 25. plt.xlim(0, 5) 26. plt.show() | |
| Ejecución    Gráfico, Gráfico de líneas, Gráfico de dispersión  Descripción generada automáticamente | Código QR del repositorio en GitHub |

|  |  |
| --- | --- |
| CODIGO TREE REGRESSION | |
| 1. # Importing the libraries 2. import numpy as np 3. import matplotlib.pyplot as plt 4. import pandas as pd 5. # Importing the dataset 6. dataset = pd.read\_csv('Position\_Salaries.csv') 7. X = dataset.iloc[:, 1:-1].values 8. y = dataset.iloc[:, -1].values 9. # Training the Decision Tree Regression model on the whole dataset 10. from sklearn.tree import DecisionTreeRegressor 11. regressor = DecisionTreeRegressor(random\_state = 0) 12. regressor.fit(X, y) 13. # Predicting a new result 14. regressor.predict([[6.5]]) 15. # Visualising the Decision Tree Regression results (higher resolution) 16. X\_grid = np.arange(min(X), max(X), 0.01) 17. X\_grid = X\_grid.reshape((len(X\_grid), 1)) 18. plt.scatter(X, y, color = 'red') 19. plt.plot(X\_grid, regressor.predict(X\_grid), color = 'blue') 20. plt.title('Truth or Bluff (Decision Tree Regression)') 21. plt.xlabel('Position level') 22. plt.ylabel('Salary') 23. plt.show() | |
| Ejecución |  |

|  |  |
| --- | --- |
| CODIGO MULTIPLE LINEAR REGRESSION | |
| 1. # Importing the libraries 2. import numpy as np 3. import matplotlib.pyplot as plt 4. import pandas as pd 5. # Importing the dataset 6. dataset = pd.read\_csv('50\_Startups.csv') 7. X = dataset.iloc[:, :-1].values 8. y = dataset.iloc[:, -1].values 9. print(X) 10. # Encoding categorical data 11. from sklearn.compose import ColumnTransformer 12. from sklearn.preprocessing import OneHotEncoder 13. ct = ColumnTransformer(transformers= [('encoder', OneHotEncoder (), [3])], 14. remainder='passthrough') 15. X = np.array(ct.fit\_transform(X)) 16. print(X) 17. # Splitting the dataset into the Training set and Test set 18. from sklearn.model\_selection import train\_test\_split 19. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, 20. random\_state = 0) 21. # Training the Multiple Linear Regression model on the Training set 22. from sklearn.linear\_model import LinearRegression 23. regressor = LinearRegression() 24. regressor.fit (X\_train, y\_train) 25. # Predicting the Test set results 26. y\_pred = regressor.predict (X\_test) 27. np.set\_printoptions(precision=2) 28. print(np.concatenate((y\_pred.reshape(len(y\_pred),1), 29. y\_test.reshape(len (y\_test),1)),1)) | |
| Ejecución |  |

|  |  |
| --- | --- |
| CODIGO POLYNOMIAL REGRESSION | |
| 1. # Importing the libraries 2. import numpy as np 3. import matplotlib.pyplot as plt 4. import pandas as pd 5. # Importing the dataset 6. dataset = pd.read\_csv('Position\_Salaries.csv') 7. X = dataset.iloc[:, 1:-1].values 8. y = dataset.iloc[:, -1].values 9. # Training the Linear Regression model on the whole dataset 10. from sklearn.linear\_model import LinearRegression 11. lin\_reg = LinearRegression() 12. lin\_reg.fit(X, y) 13. # Training the Polynomial Regression model on the whole dataset 14. from sklearn.preprocessing import PolynomialFeatures 15. poly\_reg = PolynomialFeatures(degree = 4) 16. X\_poly = poly\_reg.fit\_transform(X) 17. lin\_reg\_2 = LinearRegression() 18. lin\_reg\_2.fit (X\_poly, y) 19. # Visualising the Linear Regression results 20. plt.scatter(X, y, color= 'red') 21. plt.plot(X, lin\_reg.predict (X), color = 'blue') 22. plt.title('Truth or Bluff (Linear Regression)') 23. plt.xlabel('Position Level') 24. plt.ylabel('Salary') 25. plt.show() 26. # Visualising the Polynomial Regression results 27. plt.scatter(X, y, color = 'red') 28. plt.plot(X, lin\_reg\_2.predict(poly\_reg.fit\_transform(X)), color = 'blue') 29. plt.title('Truth or Bluff (Polynomial Regression)') 30. plt.xlabel('Position level') 31. plt.ylabel ('Salary') 32. plt.show 33. # Visualising the Polynomial Regression results (for higher resolution and smoother curve) 34. X\_grid = np.arange(min(X), max(X), 0.1) 35. X\_grid = X\_grid.reshape((len(X\_grid), 1)) 36. plt.scatter(X, y, color = 'red') 37. plt.plot(X\_grid, lin\_reg\_2.predict(poly\_reg.fit\_transform(X\_grid)), color = 'blue') 38. plt.title('Truth or Bluff (Polynomial Regression)') 39. plt.xlabel('Position level') 40. plt.ylabel('Salary') 41. plt.show() 42. # Predicting a new result with Linear Regression 43. lin\_reg.predict([[6.5]]) 44. # Predicting a new result with Polynomial Regression 45. lin\_reg\_2.predict(poly\_reg.fit\_transform([[6.5]])) | |
| Ejecución |  |

|  |  |
| --- | --- |
| CODIGO SIMPLE LINEAR REGRESSION | |
| 1. # Simple Linear Regression 2. # Importing the libraries 3. import numpy as np 4. import matplotlib.pyplot as plt 5. import pandas as pd 6. # Importing the dataset 7. dataset = pd.read\_csv('Salary\_Data.csv') 8. X = dataset.iloc[:, :-1].values 9. y = dataset.iloc[:, -1]. Values 10. # Splitting the dataset into the Training set and Test set 11. from sklearn.model\_selection import train\_test\_split 12. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 1/3, random\_state = 0) 13. # Training the Simple Linear Regression model on the Training set 14. from sklearn.linear\_model import LinearRegression 15. regressor = LinearRegression () 16. regressor.fit(X\_train, y\_train) 17. # Predicting the Test set results 18. y\_pred = regressor.predict (X\_test) 19. # Visualising the Training set results 20. plt.scatter (X\_train, y\_train, color = 'red') 21. plt.plot(X\_train, regressor.predict ( X\_train), color='blue') 22. plt.title( 'Salary vs Experience (Training set)') 23. plt.xlabel('Years of Experience') 24. plt.ylabel('Salary') 25. plt.show() 26. # Visualising the Test set results 27. plt.scatter (X\_test, y\_test, color = 'red') 28. plt.plot(X\_train, regressor.predict (X\_train), color = 'blue') 29. plt.title( 'Salary vs Experience (Test set) ') 30. plt.xlabel ('Years of Experience') 31. plt.ylabel ('Salary') 32. plt.show() | |
| Ejecución  Gráfico, Gráfico de dispersión  Descripción generada automáticamente Gráfico, Gráfico de dispersión  Descripción generada automáticamente |  |

|  |  |
| --- | --- |
| CODIGO SUPPORT VECTOR REGRESSION (SVR) | |
| 1. # Support Vector Regression (SVR) 2. # Importing the libraries 3. import numpy as np 4. import matplotlib.pyplot as plt 5. import pandas as pd 6. # Importing the dataset 7. dataset = pd.read\_csv('Position\_Salaries.csv') 8. X = dataset.iloc[:, 1:-1].values 9. y = dataset.iloc[:, -1].values 10. print(X) 11. print (y) 12. y = y. reshape (len (y), 1) 13. print (y) 14. # Feature Scaling 15. from sklearn.preprocessing import StandardScaler 16. sc\_X = StandardScaler() 17. sc\_y = StandardScaler() 18. X = sc\_X.fit\_transform(X) 19. y = sc\_y.fit\_transform(y) 20. print(X) 21. print(y) 22. # Training the SVR model on the whole dataset 23. from sklearn.svm import SVR 24. regressor = SVR(kernel = 'rbf') 25. regressor.fit(X, y) 26. # Predicting a new result 27. sc\_y.inverse\_transform(regressor.predict(sc\_X.transform([[6.5]])).reshape(- 1, 1) ) 28. # Visualising the SVR results 29. plt.scatter(sc\_X.inverse\_transform(X), sc\_y.inverse\_transform(y),color= 'red') 30. plt.plot(sc\_X.inverse\_transform(X), 31. sc\_y.inverse\_transform(regressor.predict(X).reshape(-1,1)),color='blue') 32. plt.title('Truth or Bluff (SVR)') 33. plt.xlabel( 'Position level') 34. plt.ylabel('Salary') 35. plt.show() 36. # Visualising the SVR results (for higher resolution and smoother curve) 37. X\_grid = np.arange(min(sc\_X.inverse\_transform(X)), 38. max(sc\_X.inverse\_transform(X)), 0.1) 39. X\_grid = X\_grid.reshape((len (X\_grid), 1)) 40. plt.scatter(sc\_X.inverse\_transform(X), sc\_y.inverse\_transform(y), color = 'red' ) 41. plt.plot(X\_grid, 42. sc\_y.inverse\_transform(regressor.predict (sc\_.transform(X\_grid)) .reshape(- 1,1)), 43. color= 'blue') 44. plt.title('Truth or Bluff (SVR) ') 45. plt.xlabel ('Position level') 46. plt.ylabel ('Salary') 47. plt.show() | |
| Ejecución  Gráfico, Gráfico de líneas  Descripción generada automáticamenteGráfico, Gráfico de líneas  Descripción generada automáticamente |  |

|  |  |
| --- | --- |
| CODIGO DATA PREPROGRESSING TOOLS | |
| 1. # Data Preprocessing Tools 2. # Importing the libraries 3. import numpy as np 4. import matplotlib.pyplot as plt 5. import pandas as pd 6. # Importing the dataset 7. dataset = pd.read\_csv ('Data.csv') 8. X = dataset.iloc[:, : -1].values 9. y = dataset.iloc[:, -1].values 10. print (X) 11. print(y) 12. # Taking care of missing data 13. from sklearn.impute import SimpleImputer 14. imputer = SimpleImputer(missing\_values=np.nan,strategy='mean') 15. imputer.fit(X[:, 1:3]) 16. X[:, 1:3] = imputer.transform(X[:, 1:3]) 17. print (X) 18. # Encoding categorical data 19. # Encoding the Independent Variable 20. from sklearn.compose import ColumnTransformer 21. from sklearn.preprocessing import OneHotEncoder 22. ct= ColumnTransformer(transformers= [('encoder', OneHotEncoder(),[0])], 23. remainder ='passthrough') 24. X = np.array (ct.fit\_transform(X)) 25. print(X) 26. # Encoding the Dependent Variable 27. from sklearn.preprocessing import LabelEncoder 28. le= LabelEncoder() 29. y = le.fittransform(y) 30. print(y) 31. # Splitting the dataset into the Training set and Test set 32. from sklearn.model\_selection import train\_test\_split 33. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, 34. random state = 1) 35. print (X\_train) 36. print(X\_test) 37. print (y\_train) 38. print (y\_test) | |
| Ejecución  Texto  Descripción generada automáticamenteTexto  Descripción generada automáticamente |  |

|  |  |
| --- | --- |
| CODIGO SUPPORT VECTOR MACHINE (SVM) | |
| 1. # Importing the libraries 2. import numpy as np 3. import matplotlib.pyplot as plt 4. import pandas as pd 5. # Importing the dataset 6. dataset = pd.read\_csv('Social\_Network\_Ads.csv') 7. X = dataset.iloc[:, :-1].values 8. y = dataset.iloc[:, -1].values 9. # Splitting the dataset into the Training set and Test set 10. from sklearn.model\_selection import train\_test\_split 11. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, 12. random\_state = 0) 13. print(X\_train) 14. print(y\_train) 15. print(X\_test) 16. print(y\_test) 17. # Feature Scaling 18. from sklearn.preprocessing import StandardScaler 19. sc = StandardScaler() 20. X\_train = sc.fit\_transform(X\_train) 21. X\_test = sc.transform(X\_test) 22. print(X\_train) 23. print(X\_test) 24. # Training the SVM model on the Training set 25. from sklearn.svm import SVC 26. classifier = SVC(kernel = 'linear', random\_state = 0) 27. classifier.fit(X\_train, y\_train) 28. # Predicting a new result 29. print(classifier.predict(sc.transform([[30,87000]]))) 30. # Predicting the Test set results 31. y\_pred = classifier.predict(X\_test) 32. print(np.concatenate((y\_pred.reshape(len(y\_pred),1), 33. y\_test.reshape(len(y\_test),1)),1)) 34. # Making the Confusion Matrix 35. from sklearn.metrics import confusion\_matrix, accuracy\_score 36. cm = confusion\_matrix(y\_test, y\_pred) 37. print(cm) 38. accuracy\_score(y\_test, y\_pred) 39. # Visualising the Training set results 40. from matplotlib.colors import ListedColormap 41. X\_set, y\_set = sc.inverse\_transform(X\_train), y\_train 42. X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = 43. X\_set[:, 0].max() + 10, step = 0.25),   np.arange(start = X\_set[:, 1].min() - 1000, stop =   1. X\_set[:, 1].max() + 1000, step = 0.25)) 2. plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), 3. X2.ravel()]).T)).reshape(X1.shape),    1. alpha = 0.75, cmap = ListedColormap(('red', 'green'))) 4. plt.xlim(X1.min(), X1.max()) 5. plt.ylim(X2.min(), X2.max()) 6. for i, j in enumerate(np.unique(y\_set)): 7. plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = 8. ListedColormap(('red', 'green'))(i), label = j) 9. plt.title('SVM (Training set)') 10. plt.xlabel('Age') 11. plt.ylabel('Estimated Salary') 12. plt.legend() 13. plt.show() 14. # Visualising the Test set results 15. from matplotlib.colors import ListedColormap 16. X\_set, y\_set = sc.inverse\_transform(X\_test), y\_test 17. X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = 18. X\_set[:, 0].max() + 10, step = 0.25),     1. np.arange(start = X\_set[:, 1].min() - 1000, stop = 19. X\_set[:, 1].max() + 1000, step = 0.25)) 20. plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), 21. X2.ravel()]).T)).reshape(X1.shape), 22. alpha = 0.75, cmap = ListedColormap(('red', 'green'))) 23. plt.xlim(X1.min(), X1.max()) 24. plt.ylim(X2.min(), X2.max()) 25. for i, j in enumerate(np.unique(y\_set)): 26. plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = 27. ListedColormap(('red', 'green'))(i), label = j) 28. plt.title('SVM (Test set)') 29. plt.xlabel('Age') 30. plt.ylabel('Estimated Salary') 31. plt.legend() 32. plt.show() | |
| Ejecución  Gráfico, Gráfico de dispersión  Descripción generada automáticamente Gráfico, Gráfico de dispersión  Descripción generada automáticamente | Código QR  Descripción generada automáticamente |

|  |  |
| --- | --- |
| CODIGO K-NEAREST NEIGHBORS (K-NN) | |
| 1. # Importing the libraries 2. import numpy as np 3. import matplotlib.pyplot as plt 4. import pandas as pd 5. # Importing the dataset 6. dataset = pd.read\_csv('Social\_Network\_Ads.csv') 7. X = dataset.iloc[:, :-1].values 8. y = dataset.iloc[:, -1].values 9. # Splitting the dataset into the Training set and Test set 10. from sklearn.model\_selection import train\_test\_split 11. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, 12. random\_state = 0) 13. print(X\_train) 14. print(y\_train) 15. print(X\_test) 16. print(y\_test) 17. # Feature Scaling 18. from sklearn.preprocessing import StandardScaler 19. sc = StandardScaler() 20. X\_train = sc.fit\_transform(X\_train) 21. X\_test = sc.transform(X\_test) 22. print(X\_train) 23. print(X\_test) 24. # Training the K-NN model on the Training set 25. from sklearn.neighbors import KNeighborsClassifier 26. classifier = KNeighborsClassifier(n\_neighbors = 5, metric = 'minkowski', p = 2) 27. classifier.fit(X\_train, y\_train) 28. # Predicting a new result 29. print(classifier.predict(sc.transform([[30,87000]]))) 30. # Predicting the Test set results 31. y\_pred = classifier.predict(X\_test) 32. print(np.concatenate((y\_pred.reshape(len(y\_pred),1), 33. y\_test.reshape(len(y\_test),1)),1)) 34. # Making the Confusion Matrix 35. from sklearn.metrics import confusion\_matrix, accuracy\_score 36. cm = confusion\_matrix(y\_test, y\_pred) 37. print(cm) 38. accuracy\_score(y\_test, y\_pred) 39. # Visualising the Training set results 40. from matplotlib.colors import ListedColormap 41. X\_set, y\_set = sc.inverse\_transform(X\_train), y\_train 42. X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = 43. X\_set[:, 0].max() + 10, step = 1),     1. np.arange(start = X\_set[:, 1].min() - 1000, stop = 44. X\_set[:, 1].max() + 1000, step = 1)) 45. plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), 46. X2.ravel()]).T)).reshape(X1. shape),     1. alpha = 0.75, cmap = ListedColormap(('red', 'green'))) 47. plt.xlim(X1.min(), X1.max()) 48. plt.ylim(X2.min(), X2.max()) 49. for i, j in enumerate(np.unique(y\_set)): 50. plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = 51. ListedColormap(('red', 'green'))(i), label = j) 52. plt.title('K-NN (Training set)') 53. plt.xlabel('Age') 54. plt.ylabel('Estimated Salary') 55. plt.legend() 56. plt.show() 57. # Visualising the Test set results 58. from matplotlib.colors import ListedColormap 59. X\_set, y\_set = sc.inverse\_transform(X\_test), y\_test 60. X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = 61. X\_set[:, 0].max() + 10, step = 1),     1. np.arange(start = X\_set[:, 1].min() - 1000, stop = 62. X\_set[:, 1].max() + 1000, step = 1)) 63. plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), 64. X2.ravel()]).T)).reshape(X1.shape), 65. alpha = 0.75, cmap = ListedColormap(('red', 'green'))) 66. plt.xlim(X1.min(), X1.max()) 67. plt.ylim(X2.min(), X2.max()) 68. for i, j in enumerate(np.unique(y\_set)): 69. plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = 70. ListedColormap(('red', 'green'))(i), label = j) 71. plt.title('K-NN (Test set)') 72. plt.xlabel('Age') 73. plt.ylabel('Estimated Salary') 74. plt.legend() 75. plt.show() | |
| Ejecución  Gráfico, Gráfico de dispersión  Descripción generada automáticamenteGráfico, Gráfico de dispersión  Descripción generada automáticamente |  |

|  |  |
| --- | --- |
| CODIGO LOGISTIC REGRESSION | |
| 1. # Importing the libraries 2. import numpy as np 3. import matplotlib.pyplot as plt 4. import pandas as pd 5. # Importing the dataset 6. dataset = pd.read\_csv('Social\_Network\_Ads.csv') 7. X = dataset.iloc[:, :-1].values 8. y = dataset.iloc[:, -1].values 9. # Splitting the dataset into the Training set and Test set 10. from sklearn.model\_selection import train\_test\_split 11. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0) 12. print (X\_train) 13. print (y\_train) 14. print (X\_test) 15. print (y\_test) 16. # Feature Scaling 17. from sklearn.preprocessing import StandardScaler 18. sc = StandardScaler() 19. X\_train = sc.fit\_transform(X\_train) 20. X\_test = sc.transform(X\_test) 21. print(X\_train) 22. print(X\_test) 23. # Training the Logistic Regression model on the Training set 24. from sklearn.linear\_model import LogisticRegression 25. classifier = LogisticRegression(random\_state = 0) 26. classifier.fit(X\_train, y\_train) 27. # Predicting a new result 28. print(classifier.predict(sc.transform([[30,87000]]))) 29. # Predicting the Test set results 30. y\_pred = classifier.predict(X\_test) 31. print(np.concatenate((y\_pred. reshape (len (y\_pred), 1), y\_test.reshape(len (y\_test),1)), 1)) 32. # Making the Confusion Matrix 33. from sklearn.metrics import confusion\_matrix, accuracy\_score 34. cm = confusion\_matrix(y\_test, y\_pred) 35. print (cm) 36. accuracy\_score (y\_test, y\_pred) 37. # Visualising the Training set results 38. from matplotlib.colors import ListedColormap 39. X\_set, y\_set = sc.inverse\_transform(X\_train), y\_train 40. X1, X2 = np.meshgrid(np.arange(start = X\_set[:, 0].min() - 10, stop = 41. X\_set[:, 0].max() + 10, step = 0.25), 42. np.arange(start = X\_set[:, 1].min() - 1000, stop = 43. X\_set[:, 1].max() + 1000, step = 0.25)) 44. plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), 45. X2.ravel()]).T)).reshape(X1.shape), 46. alpha = 0.75, cmap = ListedColormap(('red', 'green'))) 47. plt.xlim(X1.min(), X1.max()) 48. plt.ylim(X2.min(), X2.max()) 49. for i, j in enumerate(np.unique(y\_set)): 50. plt.scatter(X\_set [y\_set == j, 0], X\_set[y\_set == j, 1], c = 51. ListedColormap(('red', 'green'))(i), label = j) 52. plt.title('Logistic Regression (Training set)') 53. plt.xlabel('Age') 54. plt.ylabel('Estimated Salary') 55. plt.legend() 56. plt.show() 57. # Visualising the Test set results 58. from matplotlib.colors import ListedColormap 59. X\_set, y\_set = sc.inverse\_transform(X\_test), y\_test 60. X1, X2 = np.meshgrid(np.arange(start= X\_set[:, 0].min() - 10, stop = 61. X\_set[:, 0].max() + 10, step = 0.25), 62. np.arange(start = X\_set[:, 1].min() - 1000, stop = 63. X\_set[:, 1].max() + 1000, step = 0.25)) 64. plt.contourf(X1, X2, classifier.predict(sc.transform(np.array([X1.ravel(), 65. X2.ravel()]).T)).reshape(X1. shape), 66. alpha = 0.75, cmap = ListedColormap(('red', 'green'))) 67. plt.xlim(X1.min(), X1.max()) 68. plt.ylim(X2.min(), X2.max()) 69. for i, j in enumerate(np.unique(y\_set)): 70. plt.scatter(X\_set[y\_set == j, 0], X\_set[y\_set == j, 1], c = 71. ListedColormap(('red','green'))(1), label = j) 72. plt.title('Logistic Regression (Test set)') 73. plt.xlabel('Age') 74. plt.ylabel('Estimated Salary') 75. plt.legend() 76. plt.show() | |
| Ejecución  Gráfico, Gráfico de dispersión  Descripción generada automáticamenteGráfico, Gráfico de dispersión  Descripción generada automáticamente |  |

|  |  |
| --- | --- |
| CODIGO ARTIFICIAL NEURAL NETWORK | |
| 1. # Artificial Neural Network 2. # Importing the libraries 3. import numpy as np 4. import pandas as pd 5. import tensorflow as tf 6. tf.\_version\_ 7. # Part 1 - Data Preprocessing 8. # Importing the dataset 9. dataset = pd.read\_csv('Churn\_Modelling.csv') 10. X = dataset.iloc[:, 3:-1].values 11. y = dataset.iloc[:, -1].values 12. print(X) 13. print(y) 14. # Encoding categorical data 15. # Label Encoding the "Gender" column 16. from sklearn.preprocessing import LabelEncoder 17. le = LabelEncoder( ) 18. X[:, 2] = le.fit\_transform(X[:, 2]) 19. print (X) 20. # One Hot Encoding the "Geography" column 21. from sklearn.compose import ColumnTransformer 22. from sklearn.preprocessing import OneHotEncoder 23. ct = ColumnTransformer (transformers=[('encoder', OneHotEncoder(), [1])], 24. remainder= 'passthrough') 25. X = np.array(ct.fit\_transform(X)) 26. print(X) 27. # Splitting the dataset into the Training set and Test set 28. from sklearn.model\_selection import train\_test\_split 29. X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.2, 30. random state = 0) 31. # Feature Scaling 32. from sklearn.preprocessing import StandardScaler 33. sc = StandardScaler() 34. X\_train = sc.fit\_transform(X\_train) 35. X\_test = sc.transform(X\_test) 36. # Part 2 - Building the ANN 37. # Initializing the ANN 38. ann = tf.keras.models.Sequential() 39. # Adding the input layer and the first hidden layer 40. ann.add(tf.keras.layers.Dense(units=6, activation='relu')) 41. # Adding the second hidden layer 42. ann.add(tf.keras.layers.Dense(units=6, activation='relu")) 43. # Adding the output layer 44. ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid')) 45. # Part 3 - Training the ANN 46. # Compiling the ANN 47. ann.compile(optimizer = 'adam', loss = 'binary \_crossentropy', metrics = 48. ['accuracy']) 49. # Training the AN on the Training set 50. ann.fit(X\_train, y\_train, batch\_size = 32, epochs = 100) 51. # Part 4 - Making the predictions and evaluating the model 52. # Predicting the result of a single observation 53. print(ann.predict(sc.transform([[1, 0, 0, 600, 1, 40, 3, 60000, 2, 1, 1, 54. 50000]])) > 0.5) 55. # Predicting the Test set results 56. y\_pred = ann.predict(X\_test) 57. y\_pred = (y\_pred > 0.5) 58. print(np.concatenate((y\_pred.reshape(len(y\_pred),1), 59. y\_test.reshape(len(y\_test),1)),1)) 60. # Making the Confusion Matrix 61. from sklearn.metrics import confusion\_matrix, accuracy\_score 62. cm = confusion\_matrix(y\_test, y\_pred) 63. print(cm) 64. accuracy\_score(y\_test, y\_pred) | |
| Ejecución |  |