{

"cells": [

{

"cell\_type": "code",

"execution\_count": null,

"id": "fd4ecb9e",

"metadata": {},

"outputs": [],

"source": [

"#import necessary libraries\n",

"import math\n",

"import numpy as np\n",

"import pandas as pd\n",

"import seaborn as sns\n",

"import matplotlib.pyplot as plt\n",

"%matplotlib inline\n",

"\n",

"import warnings\n",

"warnings.filterwarnings('ignore')\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "1aa9ba6c",

"metadata": {},

"outputs": [],

"source": [

"#import necessary model libraries:\n",

"\n",

"from sklearn.preprocessing import StandardScaler\n",

"from statsmodels.stats.outliers\_influence import variance\_inflation\_factor\n",

"\n",

"from sklearn.linear\_model import LinearRegression, Lasso, Ridge\n",

"from sklearn.neighbors import KNeighborsRegressor\n",

"from sklearn.ensemble import RandomForestRegressor\n",

"from sklearn.ensemble import GradientBoostingRegressor\n",

"\n",

"from sklearn.model\_selection import train\_test\_split, cross\_val\_score\n",

"from sklearn.model\_selection import GridSearchCV \n",

"from sklearn.model\_selection import RandomizedSearchCV \n",

"\n",

"from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "e7556ebe",

"metadata": {},

"outputs": [],

"source": [

"#get the dataset into dataframe\n",

"df = pd.read\_csv(\"https://raw.githubusercontent.com/dsrscientist/DSData/master/Advertising.csv\")\n",

"df.head()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "6e64b308",

"metadata": {},

"outputs": [],

"source": [

"#checking the rows and columns\n",

"df.shape\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "571c613a",

"metadata": {},

"outputs": [],

"source": [

"#checking for information of independent variables\n",

"df.info()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "d870b796",

"metadata": {},

"outputs": [],

"source": [

"#checking for null values\n",

"df.isnull().sum()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "24465a67",

"metadata": {},

"outputs": [],

"source": [

"#missing values can be visualized using graphs\n",

"plt.figure()\n",

"sns.heatmap(df.isnull(),)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "a47569f7",

"metadata": {},

"outputs": [],

"source": [

"#checking the data statistics\n",

"df.describe()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "4bab4344",

"metadata": {},

"outputs": [],

"source": [

"#removing the 'Unnamed: 0' column from dataset permanently\n",

"df.drop('Unnamed: 0',axis=1,inplace=True)\n",

"df"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "41dacacb",

"metadata": {},

"outputs": [],

"source": [

"#plotting the relationship between individual features with target variables\n",

"#analysing the relation between dependent and independent variables\n",

"\n",

"plt.figure(figsize=(8,8), facecolor='white')\n",

"sns.scatterplot(x=df['TV'],y=df['sales'],color ='red', )\n",

"\n",

"sns.scatterplot(x=df['radio'],y=df['sales'],color ='blue')\n",

"\n",

"sns.scatterplot(x=df['newspaper'],y=df['sales'],color ='green')\n",

"\n",

"\n",

"plt.legend(df.columns)\n",

"plt.xlabel(\"TV, Newspaper, Radio\")\n",

"plt.ylabel(\"Sales\")\n",

"plt.title(\"Graph of Independent Variables v/s Sales\")"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "d94b56b9",

"metadata": {},

"outputs": [],

"source": [

"#checking the distribution of data over the columns\n",

"plt.figure(figsize=(8,10), facecolor='white')\n",

"ax = plt.subplot(2,1,1) \n",

"sns.distplot(df['TV'])\n",

"\n",

"ax = plt.subplot(2,1,2)\n",

"sns.boxplot(df['TV'],orient='v')\n",

" \n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "b7e76e65",

"metadata": {},

"outputs": [],

"source": [

"plt.figure(figsize=(8,10), facecolor='white')\n",

"ax = plt.subplot(2,1,1) \n",

"sns.distplot(df['newspaper'])\n",

"\n",

"ax = plt.subplot(2,1,2)\n",

"sns.boxplot(df['newspaper'],orient='v')\n",

" \n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "e13bab7b",

"metadata": {},

"outputs": [],

"source": [

"plt.figure(figsize=(8,10), facecolor='white')\n",

"ax = plt.subplot(2,1,1) \n",

"sns.distplot(df['radio'])\n",

"\n",

"ax = plt.subplot(2,1,2)\n",

"sns.boxplot(df['radio'],orient='v')\n",

" \n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "44731c0f",

"metadata": {},

"outputs": [],

"source": [

"plt.figure(figsize=(8,10), facecolor='white')\n",

"sns.pairplot(df)\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "1675b136",

"metadata": {},

"outputs": [],

"source": [

"#Checking for Outliers using visualization plots\n",

"plt.figure(figsize=(6,6), facecolor='blue')\n",

"sns.boxplot(data=df, palette='Set1')\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "92162627",

"metadata": {},

"outputs": [],

"source": [

"#let us separate independent and dependent variables\n",

"X = df.drop(columns='sales')\n",

"y = df['sales']"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "c7a87861",

"metadata": {},

"outputs": [],

"source": [

"#standardizing the data\n",

"scaler = StandardScaler()\n",

"X\_scaled = scaler.fit\_transform(X)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "aa964014",

"metadata": {},

"outputs": [],

"source": [

"#checking for the multicollinearity between the independent features\n",

"yel =pd.DataFrame()\n",

"yel['Features'] = X.columns\n",

"yel['yel'] = [variance\_inflation\_factor(X\_scaled,i) for i in range(X\_scaled.shape[1])]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "0e48e660",

"metadata": {},

"outputs": [],

"source": [

"yel"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "ec084519",

"metadata": {},

"outputs": [],

"source": [

"#generating the heatmaps to visualize multicollinearity\n",

"corr = X.corr()\n",

"plt.figure(figsize=(6,6), facecolor='white')\n",

"sns.heatmap(corr, annot=True,cmap='YlGnBu')\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "778b4f80",

"metadata": {},

"outputs": [],

"source": [

"df.skew()\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "6a45c4cd",

"metadata": {},

"outputs": [],

"source": [

"plt.figure(figsize=(15,5))\n",

"ax = plt.subplot(1,4,1)\n",

"sns.distplot(df['TV'],color='red')\n",

"ax = plt.subplot(1,4,2)\n",

"sns.distplot(df['newspaper'],color='blue')\n",

"ax = plt.subplot(1,4,3)\n",

"sns.distplot(df['radio'],color='purple')\n",

"ax = plt.subplot(1,4,4)\n",

"sns.distplot(df['sales'],color='green')\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "7f71eed5",

"metadata": {},

"outputs": [],

"source": [

"#applying log transformation\n",

"df['newspaper']=np.log(df['newspaper'])"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "6d7cbbeb",

"metadata": {},

"outputs": [],

"source": [

"#visualizing the data after reducing the skewness\n",

"plt.figure(figsize=(15,5))\n",

"ax = plt.subplot(1,4,1)\n",

"sns.distplot(df['TV'],color='red')\n",

"ax = plt.subplot(1,4,2)\n",

"sns.distplot(df['newspaper'],color='blue')\n",

"ax = plt.subplot(1,4,3)\n",

"sns.distplot(df['radio'],color='purple')\n",

"ax = plt.subplot(1,4,4)\n",

"sns.distplot(df['sales'],color='green')\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "26691320",

"metadata": {},

"outputs": [],

"source": [

"# Linear Regression:\n",

"#Splitting the data set into train test datasets and Selecting the best random state to get maximum accuracy.\n",

"m\_acc=0\n",

"m\_RS=0\n",

"for i in range(1,100):\n",

" X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.25, random\_state=i)\n",

" lr= LinearRegression()\n",

" lr.fit(X\_train,y\_train)\n",

" pred = lr.predict(X\_test)\n",

" acc = r2\_score(y\_test, pred)\n",

" if acc > m\_acc:\n",

" m\_acc= acc\n",

" m\_RS=i\n",

"print(\"The max accuracy is\",m\_acc, 'seen for random state:',i)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "d6d781d2",

"metadata": {},

"outputs": [],

"source": [

"#selecting 99 as randomstate as we got best result\n",

"X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.25, random\_state=99)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "9f124882",

"metadata": {},

"outputs": [],

"source": [

"lm = LinearRegression()\n",

"lm.fit(X\_train, y\_train)\n",

"y\_pred = lm.predict(X\_test)\n",

"\n",

"print(\"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Results\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")\n",

"print('The r2 score is:', r2\_score(y\_test, y\_pred))\n",

"print('The mean absolute error', mean\_absolute\_error(y\_test, y\_pred))\n",

"print('The mean squared error', mean\_squared\_error(y\_test, y\_pred))\n",

"print('root mean square error', math.sqrt(mean\_squared\_error(y\_test, y\_pred)))\n",

"cv = cross\_val\_score(lm, X,y,cv=5)\n",

"print('The cross validation score', cv.mean())\n",

"\n",

"print(\"\\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*XXXXXXXXXXX\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "7ecbf639",

"metadata": {},

"outputs": [],

"source": [

"#applying GridsearchCV to improve accuracy\n",

"par\_grid = {'fit\_intercept':[True,False], 'normalize':[True,False], 'copy\_X':[True, False]}\n",

"grid\_lm = GridSearchCV(estimator=lm, param\_grid=par\_grid, cv=5,n\_jobs=1, verbose=1)\n",

"\n",

"grid\_lm.fit(X\_train, y\_train)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "295eeaee",

"metadata": {},

"outputs": [],

"source": [

"#predicting the results\n",

"print(\"The best estimators:\", grid\_lm.best\_estimator\_)\n",

"print(\"The best score:\", grid\_lm.best\_score\_)\n",

"print(\"The best parameters:\", grid\_lm.best\_params\_)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "895cf04c",

"metadata": {},

"outputs": [],

"source": [

"#RE INSTANTIATING WITH BEST PARAMETERS\n",

"grid\_lm = LinearRegression(copy\_X=True, fit\_intercept=True, normalize=True)\n",

"grid\_lm.fit(X\_train, y\_train)\n",

"y\_pred1 = lm.predict(X\_test)\n",

"\n",

"print(\"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Results\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")\n",

"print('The r2 score is:', r2\_score(y\_test, y\_pred1))\n",

"print('The mean absolute error', mean\_absolute\_error(y\_test, y\_pred1))\n",

"print('The mean squared error', mean\_squared\_error(y\_test, y\_pred1))\n",

"print('root mean square error', math.sqrt(mean\_squared\_error(y\_test, y\_pred1)))\n",

"cv = cross\_val\_score(grid\_lm, X,y,cv=5)\n",

"print('The cross validation score', cv.mean())\n",

"\n",

"print(\"\\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*XXXXXXXXXXX\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "783aa9fd",

"metadata": {},

"outputs": [],

"source": [

"# Lasso Regularization-USING LASSO REGULARIZATION\n",

"lasso = Lasso()\n",

"lasso.fit(X\_train,y\_train)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "3c1dff5a",

"metadata": {},

"outputs": [],

"source": [

"pred = lasso.predict(X\_test)\n",

"\n",

"print(\"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Results\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")\n",

"print('The r2 score is:', r2\_score(y\_test, pred))\n",

"print('The mean absolute error', mean\_absolute\_error(y\_test, pred))\n",

"print('The mean squared error', mean\_squared\_error(y\_test, pred))\n",

"print('root mean square error', math.sqrt(mean\_squared\_error(y\_test, pred)))\n",

"cv = cross\_val\_score(lasso, X,y,cv=5)\n",

"print('The cross validation score', cv.mean())\n",

"\n",

"print(\"\\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*XXXXXXXXXXX\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "5d17f0a6",

"metadata": {},

"outputs": [],

"source": [

"#applying GridsearchCV to improve accuracy\n",

"param = {'alpha': np.arange(0.0001,0.1,0.001)}\n",

"grid\_lass= GridSearchCV(estimator=lasso,param\_grid=param,n\_jobs=2,cv=5,verbose=2)\n",

"\n",

"grid\_lass.fit(X\_train,y\_train)\n",

"pred1 = grid\_lass.predict(X\_test)\n",

"\n",

"print(\"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Results\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")\n",

"print('The r2 score is:', r2\_score(y\_test, pred1))\n",

"print(\"The best score:\", grid\_lass.best\_score\_)\n",

"\n",

"print(\"\\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*XXXXXXXXXXX\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "01f18de7",

"metadata": {},

"outputs": [],

"source": [

"#KNN Algorithm\n",

"knn\_reg = KNeighborsRegressor()\n",

"knn\_reg.fit(X\_train,y\_train)\n",

"y\_pred = knn\_reg.predict(X\_test)\n",

"\n",

"print(\"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Results\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")\n",

"print('The r2 score is:', r2\_score(y\_test, y\_pred))\n",

"print('The mean absolute error', mean\_absolute\_error(y\_test, y\_pred))\n",

"print('The mean squared error', mean\_squared\_error(y\_test, y\_pred))\n",

"print('root mean square error', math.sqrt(mean\_squared\_error(y\_test, y\_pred)))\n",

"cv = cross\_val\_score(knn\_reg, X,y,cv=5)\n",

"print('The cross validation score', cv.mean())\n",

"\n",

"print(\"\\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*XXXXXXXXXXX\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "11d32f0c",

"metadata": {},

"outputs": [],

"source": [

"#applying GridsearchCV to improve accuracy\n",

"param = {'algorithm':['kd\_tree'], \n",

" 'n\_neighbors':[3,2,4,6,8,10,14,7,11]}\n",

"grid\_knn = GridSearchCV(estimator=knn\_reg, param\_grid=param)\n",

"grid\_knn.fit(X\_train,y\_train)\n",

"\n",

"grid\_knn.fit(X\_train,y\_train)\n",

"\n",

"pred1 = grid\_knn.predict(X\_test)\n",

"\n",

"print(\"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Results\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")\n",

"print('The r2 score is:', r2\_score(y\_test, pred1))\n",

"print(\"The best score:\", grid\_knn.best\_score\_)\n",

"\n",

"print(\"\\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*XXXXXXXXXXX\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "f2653f1d",

"metadata": {},

"outputs": [],

"source": [

"#Random Forest Regressor\n",

"random\_reg = RandomForestRegressor()\n",

"random\_reg.fit(X\_train,y\_train)\n",

"y\_pred = random\_reg.predict(X\_test)\n",

"\n",

"print(\"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Results\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")\n",

"print('The r2 score is:', r2\_score(y\_test, y\_pred))\n",

"print('The mean absolute error', mean\_absolute\_error(y\_test, y\_pred))\n",

"print('The mean squared error', mean\_squared\_error(y\_test, y\_pred))\n",

"print('root mean square error', math.sqrt(mean\_squared\_error(y\_test, y\_pred)))\n",

"cv = cross\_val\_score(random\_reg, X,y,cv=5)\n",

"print('The cross validation score', cv.mean())\n",

"\n",

"print(\"\\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*XXXXXXXXXXX\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "7b59f6b9",

"metadata": {},

"outputs": [],

"source": [

"#applying RandomsearchCV to improve accuracy\n",

"#hyperparameter \n",

"n\_estimator = [int(x) for x in np.linspace(start=10, stop=120,num=10)]\n",

"\n",

"#number of features to be considered at each split\n",

"max\_features = ['auto','sqrt']\n",

"\n",

"#max number of tree\n",

"max\_depth=[int(x) for x in np.linspace(5,50,num=6)]\n",

"\n",

"#minimum number of samples for split\n",

"min\_samples\_split = [2,5,3,7,8,4]\n",

"\n",

"#min number of samples for leaf split\n",

"min\_samples\_leaf = [1,3,2,5,7,8,4,12,15,17,9,20]\n",

"\n",

"param = {'n\_estimators':n\_estimator, 'max\_features':max\_features, 'max\_depth':max\_depth, 'min\_samples\_leaf':min\_samples\_leaf, 'min\_samples\_split':min\_samples\_split}"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "e8da09d1",

"metadata": {},

"outputs": [],

"source": [

"random\_cv = RandomizedSearchCV(estimator=random\_reg, param\_distributions=param, n\_iter=4, cv=5, n\_jobs=2,verbose=2)\n",

"random\_cv.fit(X\_train, y\_train)\n",

"pred = random\_cv.predict(X\_test)\n",

"\n",

"\n",

"print(\"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Results\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")\n",

"print('The r2 score is:', r2\_score(y\_test, pred))\n",

"print(\"The best score:\", random\_cv.best\_score\_)\n",

"\n",

"print(\"\\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*XXXXXXXXXXX\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "0d955b5f",

"metadata": {},

"outputs": [],

"source": [

"#Gradient Boosting Regressor\n",

"gb\_reg = GradientBoostingRegressor()\n",

"gb\_reg.fit(X\_train,y\_train)\n",

"y\_pred = gb\_reg.predict(X\_test)\n",

"\n",

"print(\"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Results\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")\n",

"print('The r2 score is:', r2\_score(y\_test, y\_pred))\n",

"print('The mean absolute error', mean\_absolute\_error(y\_test, y\_pred))\n",

"print('The mean squared error', mean\_squared\_error(y\_test, y\_pred))\n",

"print('root mean square error', math.sqrt(mean\_squared\_error(y\_test, y\_pred)))\n",

"cv = cross\_val\_score(gb\_reg, X,y,cv=5)\n",

"print('The cross validation score', cv.mean())\n",

"\n",

"print(\"\\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*XXXXXXXXXXX\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "1dd374b9",

"metadata": {},

"outputs": [],

"source": [

"#applying RandomsearchCV to improve accuracy\n",

"\n",

"params = {\"n\_estimators\":[50,100,200,300,400,500,600,700,800,900],\"max\_depth\":[3,4,5,6,7,8,9,10,12,15],\"min\_samples\_split\":[2,5,8,10,12,15,18,20,22],\n",

" \"max\_features\":['auto','sqrt'],\"min\_samples\_leaf\":[1,3,5,6,7,8],\"learning\_rate\":[0.01,0.05,0.1,0.3,0.5,0.6,0.7]}\n",

"\n",

"random\_gb = RandomizedSearchCV(gb\_reg,param\_distributions=params,n\_iter=30,n\_jobs=2,cv=6,verbose=2)\n",

"random\_gb.fit(X\_train,y\_train)\n",

" \n",

"pred = random\_gb.predict(X\_test)\n",

"\n",

"\n",

"print(\"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Results\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")\n",

"print('The r2 score is:', r2\_score(y\_test, pred))\n",

"print(\"The best score:\", random\_gb.best\_score\_)\n",

"\n",

"print(\"\\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*XXXXXXXXXXX\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\")"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "85d08392",

"metadata": {},

"outputs": [],

"source": [

"#BEST MODEL\n",

"y\_pred = gb\_reg.predict(X\_test)\n",

"print(\"The accuracy score of model is:\",r2\_score(y\_test, y\_pred)\*100)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "84341d38",

"metadata": {},

"outputs": [],

"source": [

"#save the model\n",

"import pickle\n",

"\n",

"#open a file where you want to store the dat\n",

"file = open('Advertising\_regressor.pkl','wb')"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "186772f7",

"metadata": {},

"outputs": [],

"source": [

"#dump the information to the file\n",

"pickle.dump(gb\_reg,file)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "dd16bd50",

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3 (ipykernel)",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.9.7"

}

},

"nbformat": 4,

"nbformat\_minor": 5

}