{

"cells": [

{

"cell\_type": "code",

"execution\_count": 2,

"id": "693dc0eb",

"metadata": {},

"outputs": [],

"source": [

"#For Data loading and preprocessing\n",

"import pandas as pd\n",

"\n",

"#For matrix operations\n",

"import numpy as np\n",

"\n",

"#For plotting\n",

"import matplotlib.pyplot as plt\n",

"%matplotlib inline\n",

"import seaborn as sns\n",

"\n",

"#For splitting the data\n",

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

"\n",

"#For data preprocessing\n",

"from sklearn.preprocessing import StandardScaler\n",

"\n",

"#For hyperparameter tuning\n",

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

"#For appling LogisticRegression\n",

"from sklearn.linear\_model import LogisticRegression\n",

"\n",

"#For model/vatiable persistence \n",

"#from sklearn.externals import joblib\n",

"#For math operations\n",

"import math \n",

"\n",

"#To see the progress of the iterations\n",

"#from tqdm import tqdm\n",

"\n",

"#Performance metrices\n",

"from sklearn.metrics import roc\_auc\_score,roc\_curve,auc,log\_loss,confusion\_matrix\n",

"\n",

"#For encoding the features\n",

"from sklearn.preprocessing import LabelEncoder,LabelBinarizer\n",

"\n",

"#For ignoring warnings\n",

"import warnings\n",

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

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"id": "c07713be",

"metadata": {},

"outputs": [],

"source": [

"df=pd.read\_csv(\"https://raw.githubusercontent.com/IBM-EPBL/IBM-Project-8468-1658920292/main/project%20development/Data%20Set/weatherAUS.csv\")"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"id": "21c5f6c6",

"metadata": {},

"outputs": [],

"source": [

"df.drop\_duplicates(inplace=True)"

]

},

{

"cell\_type": "code",

"execution\_count": 6,

"id": "4523e205",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"Date False\n",

"Location False\n",

"MinTemp True\n",

"MaxTemp True\n",

"Rainfall True\n",

"Evaporation True\n",

"Sunshine True\n",

"WindGustDir True\n",

"WindGustSpeed True\n",

"WindDir9am True\n",

"WindDir3pm True\n",

"WindSpeed9am True\n",

"WindSpeed3pm True\n",

"Humidity9am True\n",

"Humidity3pm True\n",

"Pressure9am True\n",

"Pressure3pm True\n",

"Cloud9am True\n",

"Cloud3pm True\n",

"Temp9am True\n",

"Temp3pm True\n",

"RainToday True\n",

"RainTomorrow True\n",

"dtype: bool"

]

},

"execution\_count": 6,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"df.isnull().any()"

]

},

{

"cell\_type": "code",

"execution\_count": 8,

"id": "1cd9ce0d",

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"sns.set(style=\"whitegrid\")\n",

"sns.countplot(df.RainTomorrow)\n",

"plt.title(\"Target labels\")\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": 9,

"id": "c629ef90",

"metadata": {},

"outputs": [],

"source": [

"#Separating the data based on its class label.\n",

"data\_yes = df[df['RainTomorrow']=='Yes']\n",

"data\_no = df[df['RainTomorrow']=='No']"

]

},

{

"cell\_type": "code",

"execution\_count": 10,

"id": "a483df1f",

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Date</th>\n",

" <th>Location</th>\n",

" <th>MinTemp</th>\n",

" <th>MaxTemp</th>\n",

" <th>Rainfall</th>\n",

" <th>Evaporation</th>\n",

" <th>Sunshine</th>\n",

" <th>WindGustDir</th>\n",

" <th>WindGustSpeed</th>\n",

" <th>WindDir9am</th>\n",

" <th>...</th>\n",

" <th>Humidity9am</th>\n",

" <th>Humidity3pm</th>\n",

" <th>Pressure9am</th>\n",

" <th>Pressure3pm</th>\n",

" <th>Cloud9am</th>\n",

" <th>Cloud3pm</th>\n",

" <th>Temp9am</th>\n",

" <th>Temp3pm</th>\n",

" <th>RainToday</th>\n",

" <th>RainTomorrow</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>2010-09-03</td>\n",

" <td>Portland</td>\n",

" <td>9.6</td>\n",

" <td>17.2</td>\n",

" <td>0.0</td>\n",

" <td>4.0</td>\n",

" <td>0.0</td>\n",

" <td>W</td>\n",

" <td>39.0</td>\n",

" <td>N</td>\n",

" <td>...</td>\n",

" <td>99.0</td>\n",

" <td>69.0</td>\n",

" <td>1014.0</td>\n",

" <td>1010.4</td>\n",

" <td>8.0</td>\n",

" <td>8.0</td>\n",

" <td>13.8</td>\n",

" <td>16.0</td>\n",

" <td>No</td>\n",

" <td>Yes</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"<p>1 rows × 23 columns</p>\n",

"</div>"

],

"text/plain": [

" Date Location MinTemp MaxTemp Rainfall Evaporation Sunshine \\\n",

"0 2010-09-03 Portland 9.6 17.2 0.0 4.0 0.0 \n",

"\n",

" WindGustDir WindGustSpeed WindDir9am ... Humidity9am Humidity3pm \\\n",

"0 W 39.0 N ... 99.0 69.0 \n",

"\n",

" Pressure9am Pressure3pm Cloud9am Cloud3pm Temp9am Temp3pm RainToday \\\n",

"0 1014.0 1010.4 8.0 8.0 13.8 16.0 No \n",

"\n",

" RainTomorrow \n",

"0 Yes \n",

"\n",

"[1 rows x 23 columns]"

]

},

"execution\_count": 10,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Observing the mode for all columns when RainTomorrow = Yes \n",

"mode\_values\_for\_yes = data\_yes.mode()\n",

"mode\_values\_for\_yes"

]

},

{

"cell\_type": "code",

"execution\_count": 11,

"id": "ff3bb89e",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"23.3"

]

},

"execution\_count": 11,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"data\_no['MaxTemp'].median()"

]

},

{

"cell\_type": "code",

"execution\_count": 12,

"id": "18ec18cc",

"metadata": {},

"outputs": [],

"source": [

"#For Temparatures we cannot replace NaN values with 0, hence replacing NaN with its respective mode value\n",

"data\_yes['MinTemp'].fillna(value=data\_yes['MinTemp'].mode()[0],inplace=True )\n",

"data\_no['MinTemp'].fillna(value=data\_no['MinTemp'].mode()[0],inplace=True )\n",

"\n",

"data\_yes['MaxTemp'].fillna(value=data\_yes['MaxTemp'].mode()[0],inplace=True )\n",

"data\_no['MaxTemp'].fillna(value=data\_no['MaxTemp'].mode()[0],inplace=True )\n",

"\n",

"\n",

"data\_yes['Temp9am'].fillna(value=data\_yes['Temp9am'].mode()[0],inplace=True )\n",

"data\_no['Temp9am'].fillna(value=data\_no['Temp9am'].mode()[0],inplace=True )\n",

"\n",

"data\_yes['Temp3pm'].fillna(value=data\_yes['Temp3pm'].mode()[0],inplace=True )\n",

"data\_no['Temp3pm'].fillna(value=data\_no['Temp3pm'].mode()[0],inplace=True )\n",

"\n",

"\n",

"# For humidity also \n",

"data\_yes['Humidity9am'].fillna(value=data\_yes['Humidity9am'].mode()[0],inplace=True )\n",

"data\_no['Humidity9am'].fillna(value=data\_no['Humidity9am'].mode()[0],inplace=True )\n",

"\n",

"\n",

"\n",

"data\_yes['Humidity3pm'].fillna(value=data\_yes['Humidity3pm'].mode()[0],inplace=True )\n",

"data\_no['Humidity3pm'].fillna(value=data\_no['Humidity3pm'].mode()[0],inplace=True )\n",

"\n",

"# For the rain fall feature we can replace NaN with 0.0 which says there is no rain fall\n",

"data\_yes['Rainfall'].fillna(value=0.0,inplace=True)\n",

"data\_no['Rainfall'].fillna(value=0.0,inplace=True)\n",

"data\_yes['Pressure9am'].fillna(value=data\_yes['Pressure9am'].median(),inplace=True )\n",

"data\_no['Pressure9am'].fillna(value=data\_no['Pressure9am'].median(),inplace=True )\n",

"\n",

"data\_yes['Pressure3pm'].fillna(value=data\_yes['Pressure3pm'].median(),inplace=True )\n",

"data\_no['Pressure3pm'].fillna(value=data\_no['Pressure3pm'].median(),inplace=True )\n",

"\n",

"\n",

"data\_yes['WindSpeed9am'].fillna(value=data\_yes['WindSpeed9am'].median(),inplace=True )\n",

"data\_no['WindSpeed9am'].fillna(value=data\_no['WindSpeed9am'].median(),inplace=True )\n",

"\n",

"data\_yes['WindSpeed3pm'].fillna(value=data\_yes['WindSpeed3pm'].median(),inplace=True )\n",

"data\_no['WindSpeed3pm'].fillna(value=data\_no['WindSpeed3pm'].median(),inplace=True )\n",

"\n",

"#WindGustSpeed -- replacing with median\n",

"data\_yes['WindGustSpeed'].fillna(value=data\_yes['WindGustSpeed'].median(),inplace=True)\n",

"data\_no['WindGustSpeed'].fillna(value=data\_no['WindGustSpeed'].median(),inplace=True)"

]

},

{

"cell\_type": "code",

"execution\_count": 13,

"id": "812e310e",

"metadata": {},

"outputs": [],

"source": [

"# For RainToday feature we cannot fill any value, so better to remove the NaN values \n",

"data\_yes.dropna(inplace=True)\n",

"data\_no.dropna(inplace=True)"

]

},

{

"cell\_type": "code",

"execution\_count": 14,

"id": "b2fe36a2",

"metadata": {},

"outputs": [],

"source": [

"data\_filled= data\_yes.append(data\_no, ignore\_index=True)"

]

},

{

"cell\_type": "code",

"execution\_count": 15,

"id": "09cee1f3",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"Date False\n",

"Location False\n",

"MinTemp False\n",

"MaxTemp False\n",

"Rainfall False\n",

"Evaporation False\n",

"Sunshine False\n",

"WindGustDir False\n",

"WindGustSpeed False\n",

"WindDir9am False\n",

"WindDir3pm False\n",

"WindSpeed9am False\n",

"WindSpeed3pm False\n",

"Humidity9am False\n",

"Humidity3pm False\n",

"Pressure9am False\n",

"Pressure3pm False\n",

"Cloud9am False\n",

"Cloud3pm False\n",

"Temp9am False\n",

"Temp3pm False\n",

"RainToday False\n",

"RainTomorrow False\n",

"dtype: bool"

]

},

"execution\_count": 15,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"data\_filled.isnull().any()"

]

},

{

"cell\_type": "code",

"execution\_count": 16,

"id": "abf707ac",

"metadata": {},

"outputs": [],

"source": [

"# sorting the data based on data (Time based splitting)\n",

"data\_filled=data\_filled.sort\_values(by='Date')"

]

},

{

"cell\_type": "code",

"execution\_count": 18,

"id": "77677b79",

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>MinTemp</th>\n",

" <th>MaxTemp</th>\n",

" <th>Rainfall</th>\n",

" <th>Evaporation</th>\n",

" <th>Sunshine</th>\n",

" <th>WindGustDir</th>\n",

" <th>WindGustSpeed</th>\n",

" <th>WindDir9am</th>\n",

" <th>WindDir3pm</th>\n",

" <th>WindSpeed9am</th>\n",

" <th>...</th>\n",

" <th>Humidity9am</th>\n",

" <th>Humidity3pm</th>\n",

" <th>Pressure9am</th>\n",

" <th>Pressure3pm</th>\n",

" <th>Cloud9am</th>\n",

" <th>Cloud3pm</th>\n",

" <th>Temp9am</th>\n",

" <th>Temp3pm</th>\n",

" <th>RainToday</th>\n",

" <th>RainTomorrow</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>3390</th>\n",

" <td>8.0</td>\n",

" <td>24.3</td>\n",

" <td>0.0</td>\n",

" <td>3.4</td>\n",

" <td>6.3</td>\n",

" <td>NW</td>\n",

" <td>30.0</td>\n",

" <td>SW</td>\n",

" <td>NW</td>\n",

" <td>6.0</td>\n",

" <td>...</td>\n",

" <td>68.0</td>\n",

" <td>29.0</td>\n",

" <td>1019.7</td>\n",

" <td>1015.0</td>\n",

" <td>7.0</td>\n",

" <td>7.0</td>\n",

" <td>14.4</td>\n",

" <td>23.6</td>\n",

" <td>No</td>\n",

" <td>Yes</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3391</th>\n",

" <td>14.0</td>\n",

" <td>26.9</td>\n",

" <td>3.6</td>\n",

" <td>4.4</td>\n",

" <td>9.7</td>\n",

" <td>ENE</td>\n",

" <td>39.0</td>\n",

" <td>E</td>\n",

" <td>W</td>\n",

" <td>4.0</td>\n",

" <td>...</td>\n",

" <td>80.0</td>\n",

" <td>36.0</td>\n",

" <td>1012.4</td>\n",

" <td>1008.4</td>\n",

" <td>5.0</td>\n",

" <td>3.0</td>\n",

" <td>17.5</td>\n",

" <td>25.7</td>\n",

" <td>Yes</td>\n",

" <td>Yes</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3392</th>\n",

" <td>13.7</td>\n",

" <td>23.4</td>\n",

" <td>3.6</td>\n",

" <td>5.8</td>\n",

" <td>3.3</td>\n",

" <td>NW</td>\n",

" <td>85.0</td>\n",

" <td>N</td>\n",

" <td>NNE</td>\n",

" <td>6.0</td>\n",

" <td>...</td>\n",

" <td>82.0</td>\n",

" <td>69.0</td>\n",

" <td>1009.5</td>\n",

" <td>1007.2</td>\n",

" <td>8.0</td>\n",

" <td>7.0</td>\n",

" <td>15.4</td>\n",

" <td>20.2</td>\n",

" <td>Yes</td>\n",

" <td>Yes</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3393</th>\n",

" <td>13.3</td>\n",

" <td>15.5</td>\n",

" <td>39.8</td>\n",

" <td>7.2</td>\n",

" <td>9.1</td>\n",

" <td>NW</td>\n",

" <td>54.0</td>\n",

" <td>WNW</td>\n",

" <td>W</td>\n",

" <td>30.0</td>\n",

" <td>...</td>\n",

" <td>62.0</td>\n",

" <td>56.0</td>\n",

" <td>1005.5</td>\n",

" <td>1007.0</td>\n",

" <td>2.0</td>\n",

" <td>7.0</td>\n",

" <td>13.5</td>\n",

" <td>14.1</td>\n",

" <td>Yes</td>\n",

" <td>Yes</td>\n",

" </tr>\n",

" <tr>\n",

" <th>23626</th>\n",

" <td>7.6</td>\n",

" <td>16.1</td>\n",

" <td>2.8</td>\n",

" <td>5.6</td>\n",

" <td>10.6</td>\n",

" <td>SSE</td>\n",

" <td>50.0</td>\n",

" <td>SSE</td>\n",

" <td>ESE</td>\n",

" <td>20.0</td>\n",

" <td>...</td>\n",

" <td>68.0</td>\n",

" <td>49.0</td>\n",

" <td>1018.3</td>\n",

" <td>1018.5</td>\n",

" <td>7.0</td>\n",

" <td>7.0</td>\n",

" <td>11.1</td>\n",

" <td>15.4</td>\n",

" <td>Yes</td>\n",

" <td>No</td>\n",

" </tr>\n",

" <tr>\n",

" <th>...</th>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>50879</th>\n",

" <td>6.3</td>\n",

" <td>17.0</td>\n",

" <td>0.0</td>\n",

" <td>1.6</td>\n",

" <td>7.9</td>\n",

" <td>E</td>\n",

" <td>26.0</td>\n",

" <td>SE</td>\n",

" <td>SE</td>\n",

" <td>4.0</td>\n",

" <td>...</td>\n",

" <td>75.0</td>\n",

" <td>49.0</td>\n",

" <td>1028.6</td>\n",

" <td>1026.0</td>\n",

" <td>1.0</td>\n",

" <td>3.0</td>\n",

" <td>11.5</td>\n",

" <td>15.6</td>\n",

" <td>No</td>\n",

" <td>No</td>\n",

" </tr>\n",

" <tr>\n",

" <th>18595</th>\n",

" <td>7.6</td>\n",

" <td>19.3</td>\n",

" <td>0.0</td>\n",

" <td>3.4</td>\n",

" <td>9.4</td>\n",

" <td>W</td>\n",

" <td>35.0</td>\n",

" <td>W</td>\n",

" <td>W</td>\n",

" <td>13.0</td>\n",

" <td>...</td>\n",

" <td>73.0</td>\n",

" <td>32.0</td>\n",

" <td>1018.6</td>\n",

" <td>1015.4</td>\n",

" <td>1.0</td>\n",

" <td>1.0</td>\n",

" <td>9.4</td>\n",

" <td>18.8</td>\n",

" <td>No</td>\n",

" <td>No</td>\n",

" </tr>\n",

" <tr>\n",

" <th>35076</th>\n",

" <td>7.9</td>\n",

" <td>13.0</td>\n",

" <td>0.0</td>\n",

" <td>2.8</td>\n",

" <td>3.8</td>\n",

" <td>NNW</td>\n",

" <td>39.0</td>\n",

" <td>N</td>\n",

" <td>N</td>\n",

" <td>15.0</td>\n",

" <td>...</td>\n",

" <td>68.0</td>\n",

" <td>69.0</td>\n",

" <td>1017.6</td>\n",

" <td>1015.3</td>\n",

" <td>7.0</td>\n",

" <td>7.0</td>\n",

" <td>9.0</td>\n",

" <td>11.7</td>\n",

" <td>No</td>\n",

" <td>No</td>\n",

" </tr>\n",

" <tr>\n",

" <th>37396</th>\n",

" <td>11.0</td>\n",

" <td>24.2</td>\n",

" <td>0.0</td>\n",

" <td>2.2</td>\n",

" <td>9.8</td>\n",

" <td>ENE</td>\n",

" <td>20.0</td>\n",

" <td>SSW</td>\n",

" <td>NNE</td>\n",

" <td>2.0</td>\n",

" <td>...</td>\n",

" <td>68.0</td>\n",

" <td>53.0</td>\n",

" <td>1020.5</td>\n",

" <td>1017.3</td>\n",

" <td>6.0</td>\n",

" <td>3.0</td>\n",

" <td>15.9</td>\n",

" <td>22.6</td>\n",

" <td>No</td>\n",

" <td>No</td>\n",

" </tr>\n",

" <tr>\n",

" <th>5086</th>\n",

" <td>8.6</td>\n",

" <td>14.3</td>\n",

" <td>0.0</td>\n",

" <td>2.8</td>\n",

" <td>3.8</td>\n",

" <td>NW</td>\n",

" <td>35.0</td>\n",

" <td>N</td>\n",

" <td>NNW</td>\n",

" <td>13.0</td>\n",

" <td>...</td>\n",

" <td>67.0</td>\n",

" <td>56.0</td>\n",

" <td>1017.4</td>\n",

" <td>1015.0</td>\n",

" <td>7.0</td>\n",

" <td>7.0</td>\n",

" <td>9.3</td>\n",

" <td>13.4</td>\n",

" <td>No</td>\n",

" <td>Yes</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"<p>56669 rows × 21 columns</p>\n",

"</div>"

],

"text/plain": [

" MinTemp MaxTemp Rainfall Evaporation Sunshine WindGustDir \\\n",

"3390 8.0 24.3 0.0 3.4 6.3 NW \n",

"3391 14.0 26.9 3.6 4.4 9.7 ENE \n",

"3392 13.7 23.4 3.6 5.8 3.3 NW \n",

"3393 13.3 15.5 39.8 7.2 9.1 NW \n",

"23626 7.6 16.1 2.8 5.6 10.6 SSE \n",

"... ... ... ... ... ... ... \n",

"50879 6.3 17.0 0.0 1.6 7.9 E \n",

"18595 7.6 19.3 0.0 3.4 9.4 W \n",

"35076 7.9 13.0 0.0 2.8 3.8 NNW \n",

"37396 11.0 24.2 0.0 2.2 9.8 ENE \n",

"5086 8.6 14.3 0.0 2.8 3.8 NW \n",

"\n",

" WindGustSpeed WindDir9am WindDir3pm WindSpeed9am ... Humidity9am \\\n",

"3390 30.0 SW NW 6.0 ... 68.0 \n",

"3391 39.0 E W 4.0 ... 80.0 \n",

"3392 85.0 N NNE 6.0 ... 82.0 \n",

"3393 54.0 WNW W 30.0 ... 62.0 \n",

"23626 50.0 SSE ESE 20.0 ... 68.0 \n",

"... ... ... ... ... ... ... \n",

"50879 26.0 SE SE 4.0 ... 75.0 \n",

"18595 35.0 W W 13.0 ... 73.0 \n",

"35076 39.0 N N 15.0 ... 68.0 \n",

"37396 20.0 SSW NNE 2.0 ... 68.0 \n",

"5086 35.0 N NNW 13.0 ... 67.0 \n",

"\n",

" Humidity3pm Pressure9am Pressure3pm Cloud9am Cloud3pm Temp9am \\\n",

"3390 29.0 1019.7 1015.0 7.0 7.0 14.4 \n",

"3391 36.0 1012.4 1008.4 5.0 3.0 17.5 \n",

"3392 69.0 1009.5 1007.2 8.0 7.0 15.4 \n",

"3393 56.0 1005.5 1007.0 2.0 7.0 13.5 \n",

"23626 49.0 1018.3 1018.5 7.0 7.0 11.1 \n",

"... ... ... ... ... ... ... \n",

"50879 49.0 1028.6 1026.0 1.0 3.0 11.5 \n",

"18595 32.0 1018.6 1015.4 1.0 1.0 9.4 \n",

"35076 69.0 1017.6 1015.3 7.0 7.0 9.0 \n",

"37396 53.0 1020.5 1017.3 6.0 3.0 15.9 \n",

"5086 56.0 1017.4 1015.0 7.0 7.0 9.3 \n",

"\n",

" Temp3pm RainToday RainTomorrow \n",

"3390 23.6 No Yes \n",

"3391 25.7 Yes Yes \n",

"3392 20.2 Yes Yes \n",

"3393 14.1 Yes Yes \n",

"23626 15.4 Yes No \n",

"... ... ... ... \n",

"50879 15.6 No No \n",

"18595 18.8 No No \n",

"35076 11.7 No No \n",

"37396 22.6 No No \n",

"5086 13.4 No Yes \n",

"\n",

"[56669 rows x 21 columns]"

]

},

"execution\_count": 18,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Removing unwanted features, RISK\_MM is same as target label hence removing with data and loaction \n",

"data\_final = data\_filled.drop(['Date', 'Location'], axis=1)\n",

"data\_final"

]

},

{

"cell\_type": "code",

"execution\_count": 19,

"id": "be1cf1d6",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"<AxesSubplot:>"

]

},

"execution\_count": 19,

"metadata": {},

"output\_type": "execute\_result"

},

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 720x432 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"#Outliers we are checking only for numerical features\n",

"sns.set(style=\"whitegrid\")\n",

"plt.figure(figsize=(10, 6))\n",

"sns.boxplot(data=data\_final[['MinTemp','MaxTemp','Temp9am','Temp3pm']])"

]

},

{

"cell\_type": "code",

"execution\_count": 20,

"id": "280db3ad",

"metadata": {},

"outputs": [],

"source": [

"data\_final= data\_final[data\_final['Humidity3pm']!=0.0]\n",

"data\_final= data\_final[data\_final['Humidity9am']!=0.0]"

]

},

{

"cell\_type": "code",

"execution\_count": 21,

"id": "dd1de70a",

"metadata": {},

"outputs": [],

"source": [

"WindGustDir\_encode = LabelEncoder()\n",

"data\_final['WindGustDir']=WindGustDir\_encode.fit\_transform(data\_final['WindGustDir'])\n",

"\n",

"WindDir9am\_encode = LabelEncoder()\n",

"data\_final['WindDir9am']=WindDir9am\_encode.fit\_transform(data\_final['WindDir9am'])\n",

"\n",

"WindDir3pm\_encode = LabelEncoder()\n",

"data\_final['WindDir3pm']=WindDir3pm\_encode.fit\_transform(data\_final['WindDir3pm'])\n",

"\n",

"RainToday\_encode = LabelEncoder()\n",

"data\_final['RainToday']=RainToday\_encode.fit\_transform(data\_final['RainToday'])\n",

"\n",

"RainTomorrow\_encode = LabelEncoder()\n",

"data\_final['RainTomorrow']=RainTomorrow\_encode.fit\_transform(data\_final[\"RainTomorrow\"])"

]

},

{

"cell\_type": "code",

"execution\_count": 22,

"id": "b9f96da5",

"metadata": {},

"outputs": [],

"source": [

"Y= data\_final['RainTomorrow']\n",

"X = data\_final.drop(['RainTomorrow'],axis=1)"

]

},

{

"cell\_type": "code",

"execution\_count": 23,

"id": "f19693f4",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"['MinTemp',\n",

" 'MaxTemp',\n",

" 'Rainfall',\n",

" 'Evaporation',\n",

" 'Sunshine',\n",

" 'WindGustDir',\n",

" 'WindGustSpeed',\n",

" 'WindDir9am',\n",

" 'WindDir3pm',\n",

" 'WindSpeed9am',\n",

" 'WindSpeed3pm',\n",

" 'Humidity9am',\n",

" 'Humidity3pm',\n",

" 'Pressure9am',\n",

" 'Pressure3pm',\n",

" 'Cloud9am',\n",

" 'Cloud3pm',\n",

" 'Temp9am',\n",

" 'Temp3pm',\n",

" 'RainToday']"

]

},

"execution\_count": 23,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"column\_names=X.columns.tolist()\n",

"column\_names"

]

},

{

"cell\_type": "code",

"execution\_count": 24,

"id": "c9fd523c",

"metadata": {},

"outputs": [],

"source": [

"X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, train\_size=0.80,shuffle=False)"

]

},

{

"cell\_type": "code",

"execution\_count": 25,

"id": "75169731",

"metadata": {},

"outputs": [],

"source": [

"scaler= StandardScaler()\n",

"X\_train = scaler.fit\_transform(X\_train)\n",

"X\_test = scaler.transform(X\_test)"

]

},

{

"cell\_type": "code",

"execution\_count": 26,

"id": "63bf9946",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"array([[ 1.92569915, 1.33403259, -0.30534402, ..., 1.71272794,\n",

" 1.31675782, -0.53364311],\n",

" [ 1.05731743, -0.23186696, -0.27715516, ..., 0.3852272 ,\n",

" -0.47041152, -0.53364311],\n",

" [ 0.31299024, 0.98924737, -0.30534402, ..., -0.01149715,\n",

" 0.83334316, -0.53364311],\n",

" ...,\n",

" [-0.850021 , -1.61100784, -0.30534402, ..., -1.40003241,\n",

" -1.61302798, -0.53364311],\n",

" [-0.36930969, -0.00201014, -0.30534402, ..., -0.347187 ,\n",

" -0.01629472, -0.53364311],\n",

" [-0.74147328, -1.42424918, -0.30534402, ..., -1.35425652,\n",

" -1.36399619, -0.53364311]])"

]

},

"execution\_count": 26,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"X\_test"

]

},

{

"cell\_type": "code",

"execution\_count": 27,

"id": "612becd4",

"metadata": {},

"outputs": [],

"source": [

"def plotErrors(k,train,cv):\n",

" \n",

" plt.plot(k, train, label='Train logloss')\n",

" plt.plot(k, cv, label='CV logloss')\n",

" plt.legend()\n",

" plt.xlabel(\"log(C)= -log(λ)\")\n",

" plt.ylabel(\"Neg\_Log Loss\")\n",

" plt.title(\"Error Plot for Train and Validation data\")\n",

" plt.grid()\n",

" plt.show()"

]

},

{

"cell\_type": "markdown",

"id": "5ff28fbe",

"metadata": {},

"source": [

"\n",

"### Modeling the data using Logisitic Regression\n",

"#### Hyper-parameter tuning"

]

},

{

"cell\_type": "code",

"execution\_count": 28,

"id": "72e8100a",

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"parameters={'C':[10\*\*-6,10\*\*-5,10\*\*-4, 10\*\*-2, 10\*\*0, 10\*\*2, 10\*\*3] }\n",

"log\_c = list(map(lambda x : float(math.log(x)),parameters['C']))\n",

"\n",

"clf\_log = LogisticRegression(penalty='l2',class\_weight='balanced')\n",

"\n",

"clf = GridSearchCV(clf\_log, parameters, cv=5, scoring='neg\_log\_loss',return\_train\_score =True)\n",

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

"\n",

"train\_loss= clf.cv\_results\_['mean\_train\_score']\n",

"cv\_loss = clf.cv\_results\_['mean\_test\_score'] \n",

"\n",

"plotErrors(k=log\_c,train=train\_loss,cv=cv\_loss)"

]

},

{

"cell\_type": "code",

"execution\_count": 29,

"id": "26ac230c",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"LogisticRegression(C=1, class\_weight='balanced')"

]

},

"execution\_count": 29,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"clf = clf.best\_estimator\_\n",

"clf"

]

},

{

"cell\_type": "code",

"execution\_count": 30,

"id": "856394dc",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"LogisticRegression(C=1, class\_weight='balanced')"

]

},

"execution\_count": 30,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#Trainig with the best value of C\n",

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

]

},

{

"cell\_type": "code",

"execution\_count": 31,

"id": "9d54958a",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Log\_loss on train data is :0.42136375023317435\n",

"Log\_loss on test data is :0.42841877762512065\n"

]

}

],

"source": [

"#Printing the log-loss for both trian and test data\n",

"train\_loss = log\_loss(y\_train, clf.predict\_proba(X\_train)[:,1])\n",

"test\_loss =log\_loss(y\_test, clf.predict\_proba(X\_test)[:,1])\n",

"\n",

"\n",

"print(\"Log\_loss on train data is :{}\".format(train\_loss))\n",

"print(\"Log\_loss on test data is :{}\".format(test\_loss))"

]

},

{

"cell\_type": "code",

"execution\_count": 32,

"id": "5a5e9a46",

"metadata": {},

"outputs": [

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

]

},

"metadata": {},

"output\_type": "display\_data"

}

],

"source": [

"#Plotting AUC \n",

"train\_fpr, train\_tpr, thresholds = roc\_curve(y\_train, clf.predict\_proba(X\_train)[:,1])\n",

"test\_fpr, test\_tpr, thresholds = roc\_curve(y\_test, clf.predict\_proba(X\_test)[:,1])\n",

"plt.plot(train\_fpr, train\_tpr, label=\"train AUC =\"+str(auc(train\_fpr, train\_tpr)))\n",

"plt.plot(test\_fpr, test\_tpr, label=\"test AUC =\"+str(auc(test\_fpr, test\_tpr)))\n",

"plt.legend()\n",

"plt.xlabel(\"FPR\")\n",

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

"plt.title(\"ROC for Train and Test data with best\_fit\")\n",

"plt.grid()\n",

"plt.show()"

]

},

{

"cell\_type": "code",

"execution\_count": 34,

"id": "2b0b75a1",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"array([-0.272718 , -0.01888115, 0.10988243, -0.06915256, -0.62728539,\n",

" 0.12245798, 0.84953389, -0.03214127, -0.03536307, -0.09534929,\n",

" -0.24119127, 0.06394845, 1.1370843 , 0.99791459, -1.41556721,\n",

" -0.09055778, 0.30180472, 0.26363375, 0.12924531, 0.17698128])"

]

},

"execution\_count": 34,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"clf.coef\_[0]"

]

},

{

"cell\_type": "code",

"execution\_count": 35,

"id": "bb2252fc",

"metadata": {},

"outputs": [],

"source": [

"feature\_weights=sorted(zip(clf.coef\_[0],column\_names),reverse = True)"

]

},

{

"cell\_type": "code",

"execution\_count": 36,

"id": "3b9f085e",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"[(1.1370843017052434, 'Humidity3pm'),\n",

" (0.997914591510364, 'Pressure9am'),\n",

" (0.8495338941550139, 'WindGustSpeed'),\n",

" (0.301804715097945, 'Cloud3pm'),\n",

" (0.26363374793340466, 'Temp9am'),\n",

" (0.17698127687170961, 'RainToday'),\n",

" (0.12924531089312957, 'Temp3pm'),\n",

" (0.12245797502584098, 'WindGustDir'),\n",

" (0.10988243259883834, 'Rainfall'),\n",

" (0.0639484471328506, 'Humidity9am'),\n",

" (-0.018881148137894623, 'MaxTemp'),\n",

" (-0.03214126563471105, 'WindDir9am'),\n",

" (-0.03536307192807428, 'WindDir3pm'),\n",

" (-0.06915256265752175, 'Evaporation'),\n",

" (-0.09055778183103383, 'Cloud9am'),\n",

" (-0.09534928517970503, 'WindSpeed9am'),\n",

" (-0.24119127011743216, 'WindSpeed3pm'),\n",

" (-0.2727180022931684, 'MinTemp'),\n",

" (-0.6272853899285065, 'Sunshine'),\n",

" (-1.4155672105270423, 'Pressure3pm')]"

]

},

"execution\_count": 36,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"feature\_weights"

]

},

{

"cell\_type": "markdown",

"id": "b6a80c18",

"metadata": {},

"source": [

"## checking various machine learning algorithms for best fit..!"

]

},

{

"cell\_type": "code",

"execution\_count": 42,

"id": "5df0a073",

"metadata": {},

"outputs": [],

"source": [

"import matplotlib.pyplot as plt\n",

"from scipy.stats import itemfreq\n",

"import seaborn as sns\n",

"from sklearn import linear\_model\n",

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

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

"from sklearn.preprocessing import StandardScaler\n",

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

"from sklearn.metrics import mean\_squared\_error, make\_scorer\n",

"#from sklearn.model\_selection import train\_test\_split\n",

"%matplotlib inline\n",

"import datetime\n",

"from datetime import date, timedelta\n",

"from sklearn.tree import DecisionTreeRegressor\n",

"from sklearn.metrics import r2\_score\n",

"from sklearn.metrics import mean\_squared\_error\n",

"from sklearn.ensemble import ExtraTreesRegressor\n",

"from sklearn.ensemble import RandomForestRegressor\n",

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

"from sklearn.ensemble import GradientBoostingRegressor\n",

"from sklearn.ensemble import AdaBoostRegressor\n",

"from sklearn.ensemble import AdaBoostClassifier"

]

},

{

"cell\_type": "markdown",

"id": "4e5a622e",

"metadata": {},

"source": [

"## (1)LinearRegression"

]

},

{

"cell\_type": "code",

"execution\_count": 63,

"id": "39b3b253",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"LinearRegression()\n",

"The RMSE IS : 0.3367277944946895\n",

"\n",

"SCORE 0.33877236544413214\n"

]

}

],

"source": [

"# Fit the linear model\n",

"model = linear\_model.LinearRegression()\n",

"results = model.fit(X\_train, y\_train)\n",

"print(results)\n",

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

"\n",

"mse = mean\_squared\_error(y\_test, y\_pred)\n",

"rmse1 = np.sqrt(mse)\n",

"print(\"The RMSE IS :\",rmse1)\n",

"r2 = r2\_score(y\_test, y\_pred)\n",

"print(\"\\nSCORE\",r2)"

]

},

{

"cell\_type": "code",

"execution\_count": 64,

"id": "0b56d494",

"metadata": {},

"outputs": [

{

"data": {

"text/html": [

"<table class=\"simpletable\">\n",

"<caption>OLS Regression Results</caption>\n",

"<tr>\n",

" <th>Dep. Variable:</th> <td>RainTomorrow</td> <th> R-squared (uncentered):</th> <td> 0.291</td> \n",

"</tr>\n",

"<tr>\n",

" <th>Model:</th> <td>OLS</td> <th> Adj. R-squared (uncentered):</th> <td> 0.291</td> \n",

"</tr>\n",

"<tr>\n",

" <th>Method:</th> <td>Least Squares</td> <th> F-statistic: </th> <td> 931.5</td> \n",

"</tr>\n",

"<tr>\n",

" <th>Date:</th> <td>Sat, 12 Nov 2022</td> <th> Prob (F-statistic):</th> <td> 0.00</td> \n",

"</tr>\n",

"<tr>\n",

" <th>Time:</th> <td>20:15:40</td> <th> Log-Likelihood: </th> <td> -22282.</td> \n",

"</tr>\n",

"<tr>\n",

" <th>No. Observations:</th> <td> 45332</td> <th> AIC: </th> <td>4.460e+04</td>\n",

"</tr>\n",

"<tr>\n",

" <th>Df Residuals:</th> <td> 45312</td> <th> BIC: </th> <td>4.478e+04</td>\n",

"</tr>\n",

"<tr>\n",

" <th>Df Model:</th> <td> 20</td> <th> </th> <td> </td> \n",

"</tr>\n",

"<tr>\n",

" <th>Covariance Type:</th> <td>nonrobust</td> <th> </th> <td> </td> \n",

"</tr>\n",

"</table>\n",

"<table class=\"simpletable\">\n",

"<tr>\n",

" <td></td> <th>coef</th> <th>std err</th> <th>t</th> <th>P>|t|</th> <th>[0.025</th> <th>0.975]</th> \n",

"</tr>\n",

"<tr>\n",

" <th>x1</th> <td> -0.0518</td> <td> 0.006</td> <td> -8.372</td> <td> 0.000</td> <td> -0.064</td> <td> -0.040</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x2</th> <td> 0.0724</td> <td> 0.013</td> <td> 5.719</td> <td> 0.000</td> <td> 0.048</td> <td> 0.097</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x3</th> <td> 0.0192</td> <td> 0.002</td> <td> 8.438</td> <td> 0.000</td> <td> 0.015</td> <td> 0.024</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x4</th> <td> 0.0082</td> <td> 0.003</td> <td> 2.887</td> <td> 0.004</td> <td> 0.003</td> <td> 0.014</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x5</th> <td> -0.1046</td> <td> 0.003</td> <td> -31.014</td> <td> 0.000</td> <td> -0.111</td> <td> -0.098</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x6</th> <td> 0.0121</td> <td> 0.003</td> <td> 4.748</td> <td> 0.000</td> <td> 0.007</td> <td> 0.017</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x7</th> <td> 0.1098</td> <td> 0.003</td> <td> 34.457</td> <td> 0.000</td> <td> 0.104</td> <td> 0.116</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x8</th> <td> -0.0104</td> <td> 0.002</td> <td> -4.723</td> <td> 0.000</td> <td> -0.015</td> <td> -0.006</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x9</th> <td> 0.0027</td> <td> 0.002</td> <td> 1.088</td> <td> 0.276</td> <td> -0.002</td> <td> 0.008</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x10</th> <td> -0.0058</td> <td> 0.003</td> <td> -2.258</td> <td> 0.024</td> <td> -0.011</td> <td> -0.001</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x11</th> <td> -0.0465</td> <td> 0.003</td> <td> -16.966</td> <td> 0.000</td> <td> -0.052</td> <td> -0.041</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x12</th> <td> -0.0134</td> <td> 0.004</td> <td> -3.400</td> <td> 0.001</td> <td> -0.021</td> <td> -0.006</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x13</th> <td> 0.1727</td> <td> 0.005</td> <td> 35.396</td> <td> 0.000</td> <td> 0.163</td> <td> 0.182</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x14</th> <td> 0.1249</td> <td> 0.009</td> <td> 14.073</td> <td> 0.000</td> <td> 0.107</td> <td> 0.142</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x15</th> <td> -0.1794</td> <td> 0.009</td> <td> -20.625</td> <td> 0.000</td> <td> -0.196</td> <td> -0.162</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x16</th> <td> -0.0198</td> <td> 0.003</td> <td> -7.058</td> <td> 0.000</td> <td> -0.025</td> <td> -0.014</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x17</th> <td> 0.0113</td> <td> 0.003</td> <td> 4.033</td> <td> 0.000</td> <td> 0.006</td> <td> 0.017</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x18</th> <td> -0.0070</td> <td> 0.009</td> <td> -0.765</td> <td> 0.444</td> <td> -0.025</td> <td> 0.011</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x19</th> <td> -0.0112</td> <td> 0.014</td> <td> -0.810</td> <td> 0.418</td> <td> -0.038</td> <td> 0.016</td>\n",

"</tr>\n",

"<tr>\n",

" <th>x20</th> <td> 0.0363</td> <td> 0.002</td> <td> 14.728</td> <td> 0.000</td> <td> 0.031</td> <td> 0.041</td>\n",

"</tr>\n",

"</table>\n",

"<table class=\"simpletable\">\n",

"<tr>\n",

" <th>Omnibus:</th> <td>4112.319</td> <th> Durbin-Watson: </th> <td> 1.326</td>\n",

"</tr>\n",

"<tr>\n",

" <th>Prob(Omnibus):</th> <td> 0.000</td> <th> Jarque-Bera (JB): </th> <td>5372.195</td>\n",

"</tr>\n",

"<tr>\n",

" <th>Skew:</th> <td> 0.790</td> <th> Prob(JB): </th> <td> 0.00</td>\n",

"</tr>\n",

"<tr>\n",

" <th>Kurtosis:</th> <td> 3.588</td> <th> Cond. No. </th> <td> 23.6</td>\n",

"</tr>\n",

"</table><br/><br/>Notes:<br/>[1] R² is computed without centering (uncentered) since the model does not contain a constant.<br/>[2] Standard Errors assume that the covariance matrix of the errors is correctly specified."

],

"text/plain": [

"<class 'statsmodels.iolib.summary.Summary'>\n",

"\"\"\"\n",

" OLS Regression Results \n",

"=======================================================================================\n",

"Dep. Variable: RainTomorrow R-squared (uncentered): 0.291\n",

"Model: OLS Adj. R-squared (uncentered): 0.291\n",

"Method: Least Squares F-statistic: 931.5\n",

"Date: Sat, 12 Nov 2022 Prob (F-statistic): 0.00\n",

"Time: 20:15:40 Log-Likelihood: -22282.\n",

"No. Observations: 45332 AIC: 4.460e+04\n",

"Df Residuals: 45312 BIC: 4.478e+04\n",

"Df Model: 20 \n",

"Covariance Type: nonrobust \n",

"==============================================================================\n",

" coef std err t P>|t| [0.025 0.975]\n",

"------------------------------------------------------------------------------\n",

"x1 -0.0518 0.006 -8.372 0.000 -0.064 -0.040\n",

"x2 0.0724 0.013 5.719 0.000 0.048 0.097\n",

"x3 0.0192 0.002 8.438 0.000 0.015 0.024\n",

"x4 0.0082 0.003 2.887 0.004 0.003 0.014\n",

"x5 -0.1046 0.003 -31.014 0.000 -0.111 -0.098\n",

"x6 0.0121 0.003 4.748 0.000 0.007 0.017\n",

"x7 0.1098 0.003 34.457 0.000 0.104 0.116\n",

"x8 -0.0104 0.002 -4.723 0.000 -0.015 -0.006\n",

"x9 0.0027 0.002 1.088 0.276 -0.002 0.008\n",

"x10 -0.0058 0.003 -2.258 0.024 -0.011 -0.001\n",

"x11 -0.0465 0.003 -16.966 0.000 -0.052 -0.041\n",

"x12 -0.0134 0.004 -3.400 0.001 -0.021 -0.006\n",

"x13 0.1727 0.005 35.396 0.000 0.163 0.182\n",

"x14 0.1249 0.009 14.073 0.000 0.107 0.142\n",

"x15 -0.1794 0.009 -20.625 0.000 -0.196 -0.162\n",

"x16 -0.0198 0.003 -7.058 0.000 -0.025 -0.014\n",

"x17 0.0113 0.003 4.033 0.000 0.006 0.017\n",

"x18 -0.0070 0.009 -0.765 0.444 -0.025 0.011\n",

"x19 -0.0112 0.014 -0.810 0.418 -0.038 0.016\n",

"x20 0.0363 0.002 14.728 0.000 0.031 0.041\n",

"==============================================================================\n",

"Omnibus: 4112.319 Durbin-Watson: 1.326\n",

"Prob(Omnibus): 0.000 Jarque-Bera (JB): 5372.195\n",

"Skew: 0.790 Prob(JB): 0.00\n",

"Kurtosis: 3.588 Cond. No. 23.6\n",

"==============================================================================\n",

"\n",

"Notes:\n",

"[1] R² is computed without centering (uncentered) since the model does not contain a constant.\n",

"[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.\n",

"\"\"\""

]

},

"execution\_count": 64,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"import statsmodels.api as sm\n",

"\n",

"model = sm.OLS(y\_train, X\_train)\n",

"results = model.fit()\n",

"# Statsmodels gives R-like statistical output\n",

"results.summary()"

]

},

{

"cell\_type": "code",

"execution\_count": 65,

"id": "46ea8ade",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"[0.36759822 0.37814022 0.37498038 0.36102705 0.3711331 ]\n",

"0.37057579486173164\n",

"0.005949386725795814\n"

]

}

],

"source": [

"#cross validation using scikitlearn\n",

"reg=linear\_model.LinearRegression()\n",

"cv\_results=cross\_val\_score(reg,X\_train,y\_train,cv=5)\n",

"print(cv\_results)\n",

"print(np.mean(cv\_results))\n",

"print(np.std(cv\_results))\n",

"\n",

"#Using cross validation of score 5"

]

},

{

"cell\_type": "code",

"execution\_count": 66,

"id": "0f41a544",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"0.3269329380313437"

]

},

"execution\_count": 66,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"#regularization\n",

"ridge = Ridge(alpha=0.1, normalize = True)\n",

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

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

"ridge.score(X\_test,y\_test)"

]

},

{

"cell\_type": "markdown",

"id": "a466ff16",

"metadata": {},

"source": [

"## (2)DecisionTree Regressor"

]

},

{

"cell\_type": "code",

"execution\_count": 67,

"id": "c49be6d8",

"metadata": {},

"outputs": [],

"source": [

"dtr=DecisionTreeRegressor(max\_depth=10,min\_samples\_leaf=5,max\_leaf\_nodes=5)"

]

},

{

"cell\_type": "code",

"execution\_count": 68,

"id": "ff1e575a",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"RMSE = 0.3561983212605041\n",

"0.2600935712982957\n"

]

}

],

"source": [

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

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

"\n",

"mse = mean\_squared\_error(y\_test, y\_pred)\n",

"rmse2 = np.sqrt(mse)\n",

"print(\"RMSE = \",rmse2)\n",

"r2 = r2\_score(y\_test, y\_pred)\n",

"print(r2)"

]

},

{

"cell\_type": "markdown",

"id": "9d610904",

"metadata": {},

"source": [

"## (3)ExtraTreesRegressor"

]

},

{

"cell\_type": "code",

"execution\_count": 69,

"id": "f91a2501",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Fitting 3 folds for each of 1 candidates, totalling 3 fits\n"

]

},

{

"data": {

"text/plain": [

"ExtraTreesRegressor(max\_depth=5, n\_estimators=10)"

]

},

"execution\_count": 69,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"etr = ExtraTreesRegressor()\n",

"\n",

"# Choose some parameter combinations to try\n",

"\n",

"parameters = {'n\_estimators': [10],\n",

" 'criterion': ['mse'],\n",

" 'max\_depth': [5], \n",

" 'min\_samples\_split': [2],\n",

" 'min\_samples\_leaf': [1]\n",

" }\n",

"#We have to use RandomForestRegressor's own scorer (which is R^2 score)\n",

"\n",

"#Determines the cross-validation splitting strategy /to specify the number of folds in a (Stratified)KFold\n",

"\n",

"grid\_obj = GridSearchCV(etr, parameters,\n",

" cv=3, \n",

" n\_jobs=-1, #Number of jobs to run in parallel\n",

" verbose=1)\n",

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

"\n",

"# Set the clf to the best combination of parameters\n",

"etr = grid\_obj.best\_estimator\_\n",

"\n",

"# Fit the best algorithm to the data. \n",

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

]

},

{

"cell\_type": "code",

"execution\_count": 70,

"id": "b777dc3c",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"RMSE = 0.34160856567041453\n",

"0.31946484034960254\n"

]

}

],

"source": [

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

"mse = mean\_squared\_error(y\_test, y\_pred)\n",

"rmse3 = np.sqrt(mse)\n",

"print(\"RMSE = \",rmse3)\n",

"r2 = r2\_score(y\_test, y\_pred)\n",

"print(r2)"

]

},

{

"cell\_type": "markdown",

"id": "d8dd321f",

"metadata": {},

"source": [

"## (4) RandomForestRegression"

]

},

{

"cell\_type": "code",

"execution\_count": 71,

"id": "42512dae",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"RMSE = 0.3307932912221132\n",

"0.3618739849480975\n"

]

}

],

"source": [

"from sklearn.ensemble import RandomForestRegressor\n",

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

"\n",

"model = RandomForestRegressor()\n",

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

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

"\n",

"mse = mean\_squared\_error(y\_test, y\_pred)\n",

"rmse4 = np.sqrt(mse)\n",

"print(\"RMSE = \",rmse4)\n",

"r2 = r2\_score(y\_test, y\_pred)\n",

"print(r2)"

]

},

{

"cell\_type": "markdown",

"id": "c232d069",

"metadata": {},

"source": [

"## (5) Gradient Boosting Regressor"

]

},

{

"cell\_type": "code",

"execution\_count": 72,

"id": "feb68c05",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"RMSE = 0.3327186129269286\n",

"0.3544241767796571\n"

]

}

],

"source": [

"modelgbr = GradientBoostingRegressor(n\_estimators=100,max\_depth=2, random\_state=42)\n",

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

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

"\n",

"mse = mean\_squared\_error(y\_test, y\_pred)\n",

"rmse5 = np.sqrt(mse)\n",

"print(\"RMSE = \",rmse5)\n",

"r2 = r2\_score(y\_test, y\_pred)\n",

"print(r2)"

]

},

{

"cell\_type": "markdown",

"id": "1fc4ddcc",

"metadata": {},

"source": [

"## (6) Ada Boosting Regressor"

]

},

{

"cell\_type": "code",

"execution\_count": 73,

"id": "18bb08aa",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"RMSE = 0.34882463850931655\n",

"0.29041019079042163\n"

]

}

],

"source": [

"regr = AdaBoostRegressor(random\_state=42, n\_estimators=100)\n",

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

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

"mse = mean\_squared\_error(y\_test, y\_pred)\n",

"rmse6 = np.sqrt(mse)\n",

"print(\"RMSE = \",rmse6)\n",

"r2 = r2\_score(y\_test, y\_pred)\n",

"print(r2)"

]

},

{

"cell\_type": "code",

"execution\_count": 74,

"id": "a6cc4fb9",

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"<BarContainer object of 6 artists>"

]

},

"execution\_count": 74,

"metadata": {},

"output\_type": "execute\_result"

},

{

"data": {

"image/png": "\n",

"text/plain": [

"<Figure size 720x360 with 1 Axes>"

]

},

"metadata": {

"needs\_background": "light"

},

"output\_type": "display\_data"

}

],

"source": [

"reg = ['LinearReg','DecisionTreeReg','Extra treeReg','RandomForestReg','GradientBoostReg','AdaBoostReg']\n",

"rmse = [rmse1,rmse2,rmse3,rmse4,rmse5,rmse6]\n",

"c = ['red', 'yellow', 'black', 'blue', 'orange','green']\n",

"plt.figure(figsize=(10, 5))\n",

"plt.bar(reg,rmse,color=c)\n",

"# we plotted using rmse value -->ie,lesser rmse value have more accuracy"

]

},

{

"cell\_type": "code",

"execution\_count": 76,

"id": "bf6ca2f4",

"metadata": {},

"outputs": [],

"source": [

"def will\_rain\_fall\_for\_this\_conditions(xq):\n",

" \n",

" xq[\"WindGustDir\"]=WindGustDir\_encode.transform([xq[\"WindGustDir\"]])\n",

" xq[\"WindDir9am\"]=WindDir9am\_encode.transform([xq[\"WindDir9am\"]])\n",

" xq[\"WindDir3pm\"]=WindDir3pm\_encode.transform([xq[\"WindDir3pm\"]])\n",

" xq[\"RainToday\"]=RainToday\_encode.transform([xq[\"RainToday\"]])\n",

" xq=np.array(list((xq.values())))\n",

" final\_xq = scaler.transform(xq.reshape(1, -1))\n",

" chance=clf.predict\_proba(final\_xq)[:,1]\n",

" if chance>=0.5:\n",

" print(\"Yes, there is a {} % chance of rain can fall on tommorow \".format(chance\*100))\n",

" else:\n",

" print(\"No, there is only {}% chance of rainfall hence we cannot expect rain on tommorow \".format(chance\*100))\n",

" print(\"Because today's Humidity at 3pm ={}%,Atmosphereic Pressure at 9am={}millibars,and Wind Gust Speed ={}km/hr, which are very good sign for rainfall\"\n",

" .format(Humidity3pm,Pressure9am,WindGustSpeed))"

]

},

{

"cell\_type": "code",

"execution\_count": 77,

"id": "623c2254",

"metadata": {},

"outputs": [],

"source": [

"#Giving one query point here\n",

"\n",

"MinTemp = 26.2\n",

"MaxTemp = 31.7\n",

"Rainfall = 2.8\n",

"Evaporation = 5.4\n",

"Sunshine = 3.5\n",

"WindGustDir = \"NNW\"\n",

"WindGustSpeed = 57\n",

"WindDir9am = \"NNW\"\n",

"WindDir3pm = \"NNW\"\n",

"WindSpeed9am = 20\n",

"WindSpeed3pm = 13\n",

"Humidity9am = 81\n",

"Humidity3pm = 95\n",

"Pressure9am = 1007.2\n",

"Pressure3pm = 1006.1\n",

"Cloud9am = 7\n",

"Cloud3pm = 8\n",

"Temp9am = 28.8\n",

"Temp3pm = 25.4\n",

"RainToday =\"Yes\""

]

},

{

"cell\_type": "code",

"execution\_count": 78,

"id": "56757161",

"metadata": {},

"outputs": [],

"source": [

"point = [MinTemp,MaxTemp,Rainfall,\n",

" Evaporation,Sunshine,WindGustDir,\n",

" WindGustSpeed,WindDir9am,WindDir3pm,\n",

" WindSpeed9am,WindSpeed3pm,Humidity9am,\n",

" Humidity3pm,Pressure9am,Pressure3pm,\n",

" Cloud9am,Cloud3pm,Temp9am,Temp3pm,RainToday]\n",

"\n",

"xq=dict()\n",

"for i,name in enumerate(column\_names):\n",

" xq[name]=point[i]"

]

},

{

"cell\_type": "code",

"execution\_count": 79,

"id": "ebff2295",

"metadata": {},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Yes, there is a [99.04083842] % chance of rain can fall on tommorow \n",

"Because today's Humidity at 3pm =95%,Atmosphereic Pressure at 9am=1007.2millibars,and Wind Gust Speed =57km/hr, which are very good sign for rainfall\n"

]

}

],

"source": [

"will\_rain\_fall\_for\_this\_conditions(xq)"

]

},

{

"cell\_type": "markdown",

"id": "73bb2a72",

"metadata": {},

"source": [

"### Yes, there is a [99.04083842] % chance of rain can fall on tommorow "

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "b2727f5f",

"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

}