Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=4KGPUDSjQUlx

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

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

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import ListedColormap
from sklearn.model_selection import train_test_split
from scipy.stats import boxcox
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.metrics import classification_report, accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

%matplotlib inline
```

```
[9] # Set the resolution of the plotted figures
    plt.rcParams['figure.dpi'] = 200

    # Configure Seaborn plot styles: Set background color and use dark grid
    sns.set(rc={'axes.facecolor': '#faded9'}, style='darkgrid')
```

```
[10] # Read dataset
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:08 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=4KGPUDSjQUlx

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
[9] # Configure Seaborn plot styles: Set background color and use dark grid
    sns.set(rc={'axes.facecolor': '#faded9'}, style='darkgrid')
```

```
# Read dataset
df = pd.read_csv('/content/heart.csv')
df
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

Next steps: Generate code with df View recommended plots

Connected to Python 3 Google Compute Engine backend

ENG IN 07:08 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=4KGPUDSjQUlx

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
# Display a concise summary of the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

```
[12] # Define the continuous features
continuous_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']

# Identify the features to be converted to object data type
features_to_convert = [feature for feature in df.columns if feature not in continuous_features]
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:08 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=4KGPUDSjQUlx

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

```
# Define the continuous features
continuous_features = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']

# Identify the features to be converted to object data type
features_to_convert = [feature for feature in df.columns if feature not in continuous_features]

# Convert the identified features to object data type
df[features_to_convert] = df[features_to_convert].astype('object')

df.dtypes
```

```
age          int64
sex         object
cp          object
trestbps     int64
chol         int64
fbs         object
restecg     object
thalach      int64
exang       object
oldpeak    float64
slope       object
ca          object
thal        object
target      object
dtype: object
```

```
[13] # Get the summary statistics for numerical variables
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:08 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=4KGPUDSjQUlx

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
[12] thal        object
     target      object
     dtype: object
```

```
# Get the summary statistics for numerical variables
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	303.0	54.366337	9.082101	29.0	47.5	55.0	61.0	77.0
trestbps	303.0	131.623762	17.538143	94.0	120.0	130.0	140.0	200.0
chol	303.0	246.264026	51.830751	126.0	211.0	240.0	274.5	564.0
thalach	303.0	149.646865	22.905161	71.0	133.5	153.0	166.0	202.0
oldpeak	303.0	1.039604	1.161075	0.0	0.0	0.8	1.6	6.2

```
[14] # Get the summary statistics for categorical variables
     df.describe(include='object')
```

	sex	cp	fbs	restecg	exang	slope	ca	thal	target
count	303	303	303	303	303	303	303	303	303
unique	2	4	2	3	2	3	5	4	2
top	1	0	0	1	0	2	0	2	1
freq	207	143	258	152	204	142	175	166	165

Connected to Python 3 Google Compute Engine backend

ENG IN 07:08 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNIeA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=4KGPUDSjQUlx

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

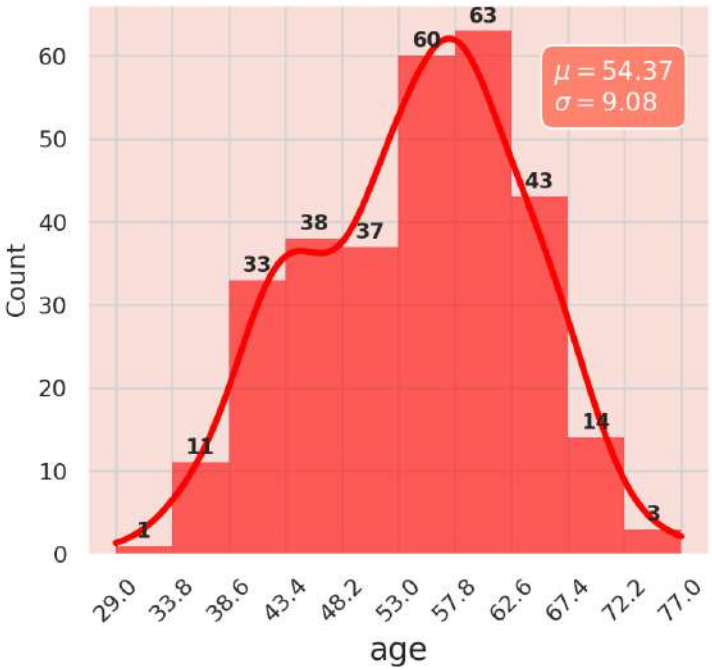
RAM Disk Colab AI

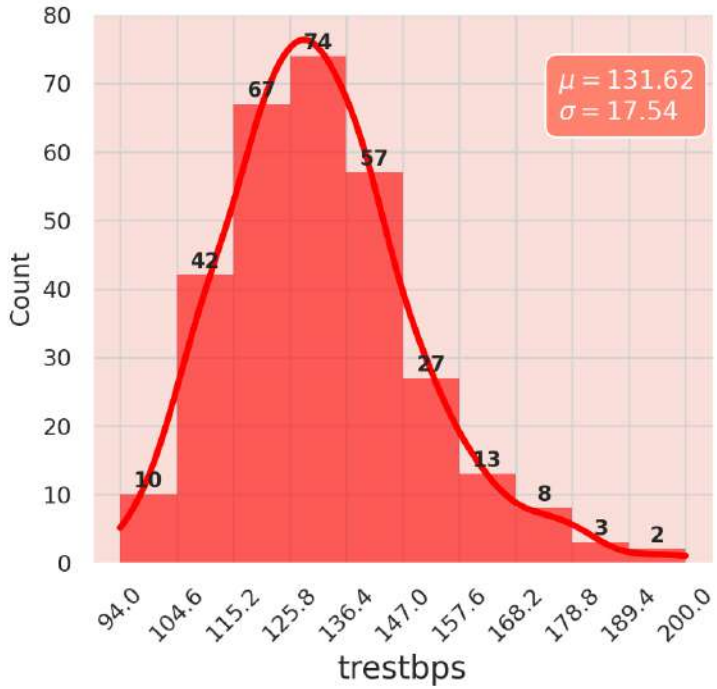
```
# Filter out continuous features for the univariate analysis
df_continuous = df[continuous_features]

# Set up the subplot
fig, ax = plt.subplots(nrows=2, ncols=3, figsize=(15, 10))

# Loop to plot histograms for each continuous feature
for i, col in enumerate(df_continuous.columns):
    x = i // 3
    y = i % 3
    values, bin_edges = np.histogram(df_continuous[col],
                                     range=(np.floor(df_continuous[col].min()), np.ceil(df_continuous[col].max())))

    graph = sns.histplot(data=df_continuous, x=col, bins=bin_edges, kde=True, ax=ax[x, y],
                         edgecolor='none', color='red', alpha=0.6, line_kws={'lw': 3})
    ax[x, y].set_xlabel(col, fontsize=15)
    ax[x, y].set_ylabel('Count', fontsize=12)
    ax[x, y].set_xticks(np.round(bin_edges, 1))
    ax[x, y].set_xticklabels(ax[x, y].get_xticks(), rotation=45)
    ax[x, y].grid(color='lightgrey')

    for j, p in enumerate(graph.patches):
        ax[x, y].annotate('{}'.format(p.get_height()), (p.get_x() + p.get_width() / 2, p.get_height() + 1),
                          ha='center', fontsize=10, fontweight="bold")

    textstr = '\n'.join((
        r'$\mu=%.2f$' % df_continuous[col].mean(),
        r'$\sigma=%.2f$' % df_continuous[col].std()
    ))
    ax[x, y].text(0.75, 0.9, textstr, transform=ax[x, y].transAxes, fontsize=12, verticalalignment='top',
                  color='white', bbox=dict(boxstyle='round', facecolor='#ff826e', edgecolor='white', pad=0.5))
```

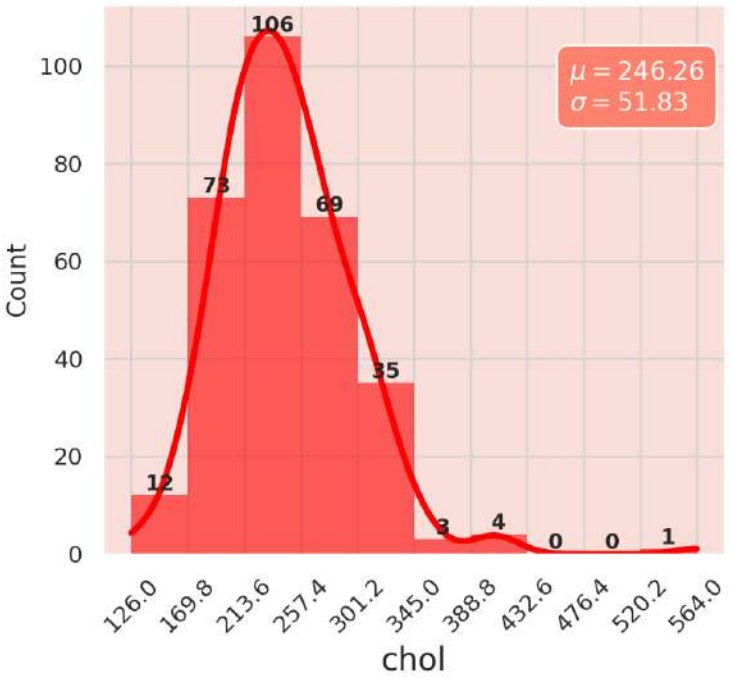
Connected to Python 3 Google Compute Engine backend

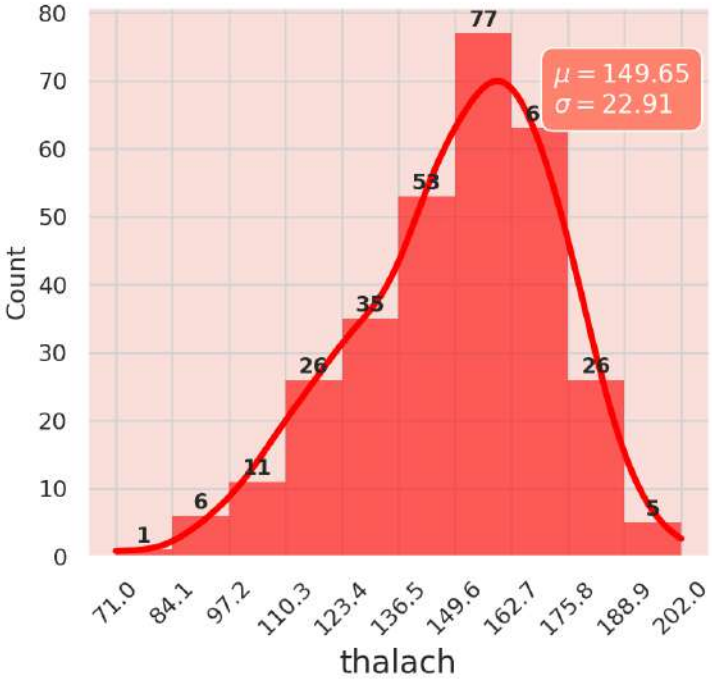
ENG IN 07:09 PM 04-04-2024

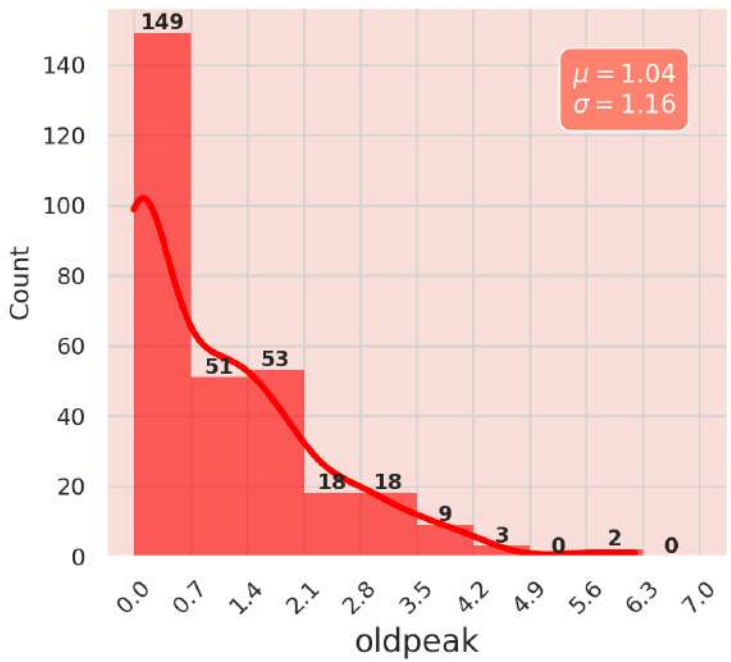
Distribution of Continuous Variables











Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=7iURMdqgQ1xu

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

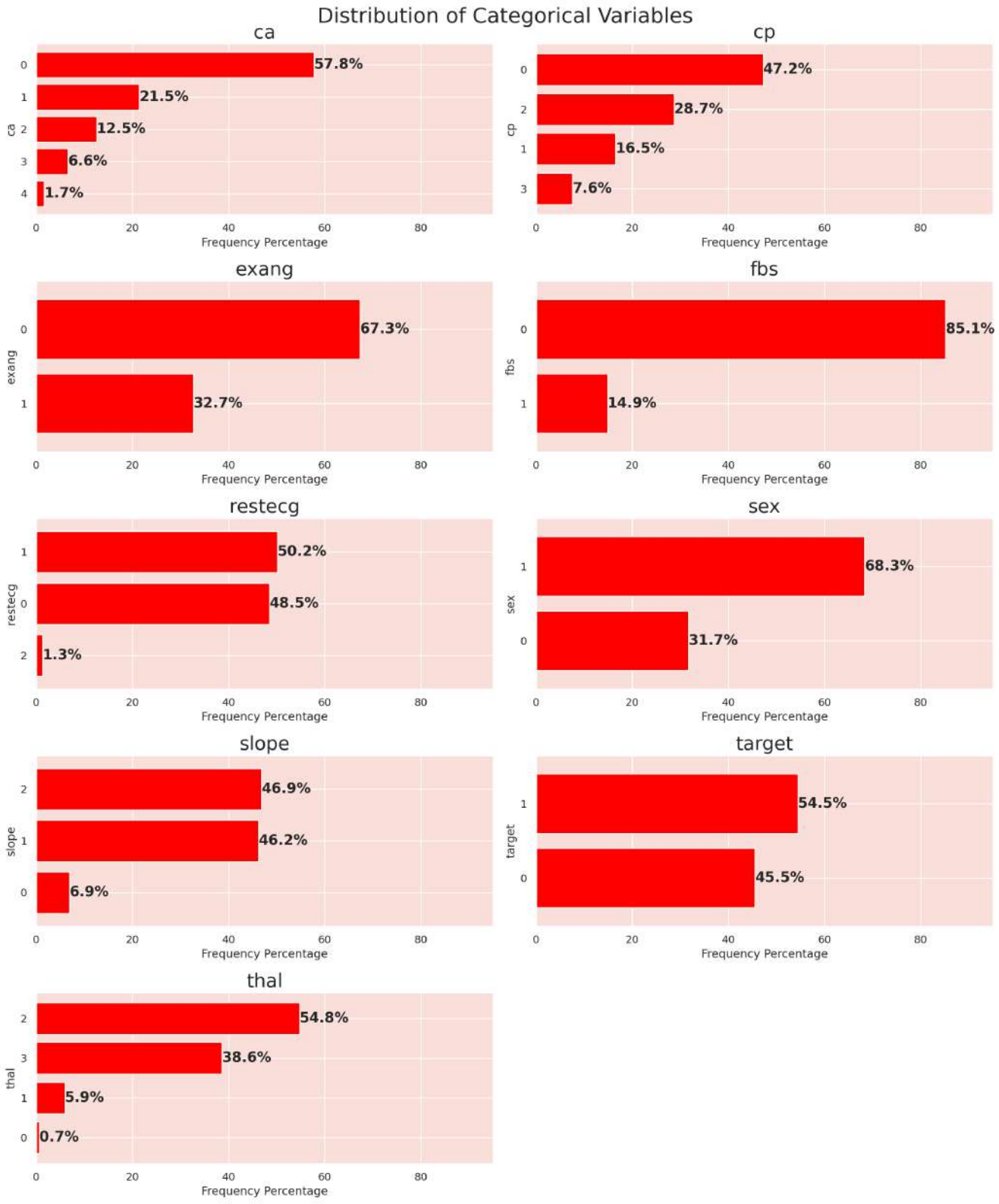
RAM Disk Colab AI

```
[16] # Filter out categorical features for the univariate analysis
     categorical_features = df.columns.difference(continuous_features)
     df_categorical = df[categorical_features]
```

```
# Set up the subplot for a 4x2 layout
fig, ax = plt.subplots(nrows=5, ncols=2, figsize=(15, 18))

# Loop to plot bar charts for each categorical feature in the 4x2 layout
for i, col in enumerate(categorical_features):
    row = i // 2
    col_idx = i % 2

    # Calculate frequency percentages
    value_counts = df[col].value_counts(normalize=True).mul(100).sort_values()

    # Plot bar chart
    value_counts.plot(kind='barh', ax=ax[row, col_idx], width=0.8, color='red')

    # Add frequency percentages to the bars
    for index, value in enumerate(value_counts):
        ax[row, col_idx].text(value, index, str(round(value, 1)) + '%', fontsize=15, weight='bold', va='center')

    ax[row, col_idx].set_xlim([0, 95])
    ax[row, col_idx].set_xlabel('Frequency Percentage', fontsize=12)
    ax[row, col_idx].set_title(f'{col}', fontsize=20)

ax[4,1].axis('off')
plt.suptitle('Distribution of Categorical Variables', fontsize=22)
plt.tight_layout()
plt.subplots_adjust(top=0.95)
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:11 PM 04-04-2024


Distribution of Categorical Variables
ca
0
1
2
3
4
57.8%
21.5%
12.5%
6.6%
1.7%
cp
0
2
1
3
47.2%
28.7%
16.5%
7.6%
exang
0
1
67.3%
32.7%
fbs
0
1
85.1%
14.9%
restecg
1
0
2
50.2%
48.5%
1.3%
sex
1
0
68.3%
31.7%
slope
2
1
0
46.9%
46.2%
6.9%
target
1
0
54.5%
45.5%
thal
2
3
1
0
54.8%
38.6%
5.9%
0.7%
0
20
40
60
80
Frequency Percentage

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=tv-dwpyNQ9bP

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

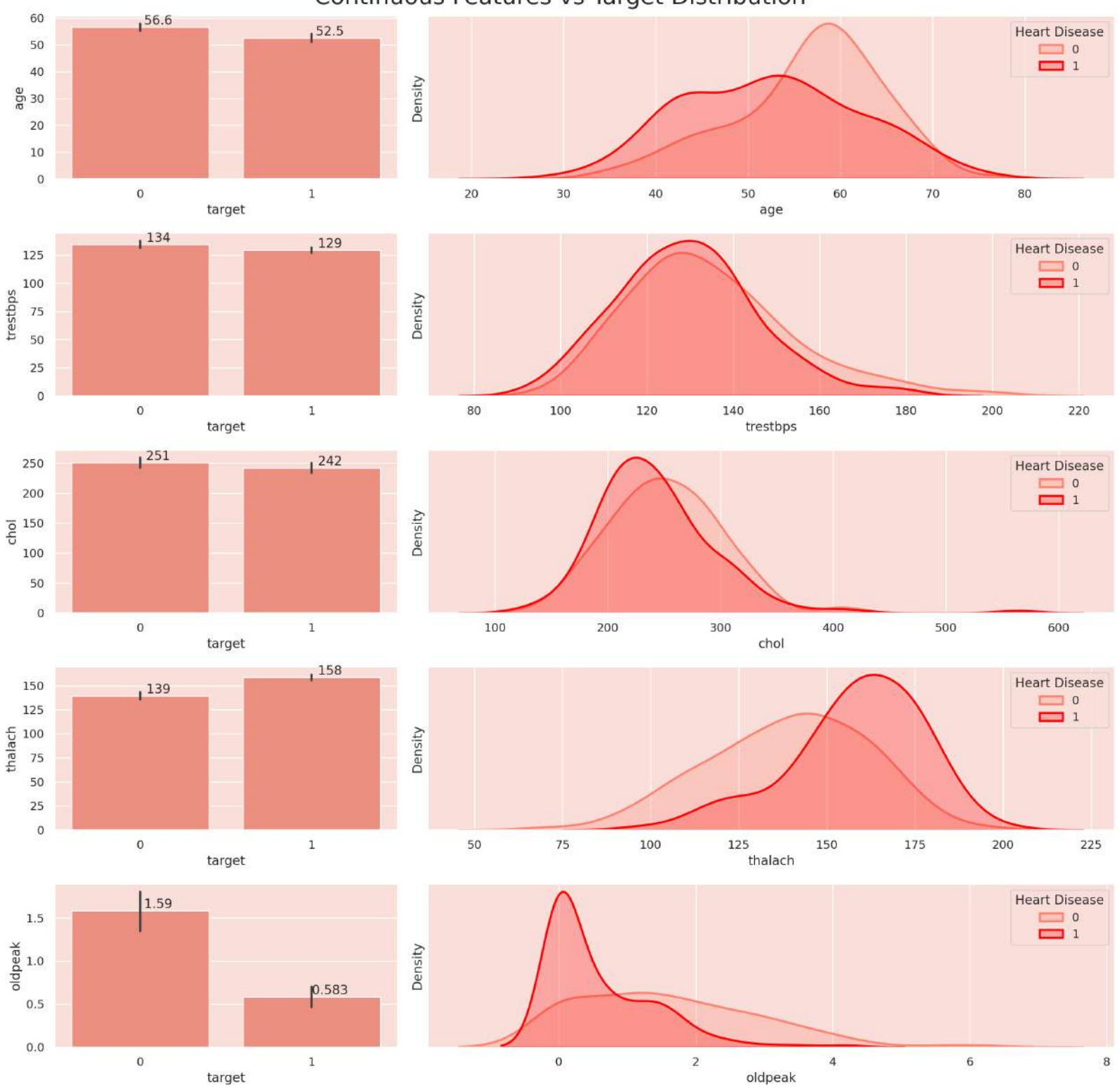
+ Code + Text

RAM Disk Colab AI

```
# Set color palette
sns.set_palette(['#ff826e', 'red'])

# Create the subplots
fig, ax = plt.subplots(len(continuous_features), 2, figsize=(15,15), gridspec_kw={'width_ratios': [1, 2]})

# Loop through each continuous feature to create barplots and kde plots
for i, col in enumerate(continuous_features):
    # Barplot showing the mean value of the feature for each target category
    graph = sns.barplot(data=df, x="target", y=col, ax=ax[i,0])

    # KDE plot showing the distribution of the feature for each target category
    sns.kdeplot(data=df[df["target"]==0], x=col, fill=True, linewidth=2, ax=ax[i,1], label='0')
    sns.kdeplot(data=df[df["target"]==1], x=col, fill=True, linewidth=2, ax=ax[i,1], label='1')
    ax[i,1].set_yticks([])
    ax[i,1].legend(title='Heart Disease', loc='upper right')

    # Add mean values to the barplot
    for cont in graph.containers:
        graph.bar_label(cont, fmt='        %.3g')

# Set the title for the entire figure
plt.suptitle('Continuous Features vs Target Distribution', fontsize=22)
plt.tight_layout()
plt.show()
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:12 PM 04-04-2024


Continuous Features vs Target Distribution
age
target
56.6
52.5
Density
Heart Disease
0
1
trestbps
134
129
chol
251
242
thalach
139
158
oldpeak
1.59
0.583

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=N5bSya2eRBgN

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

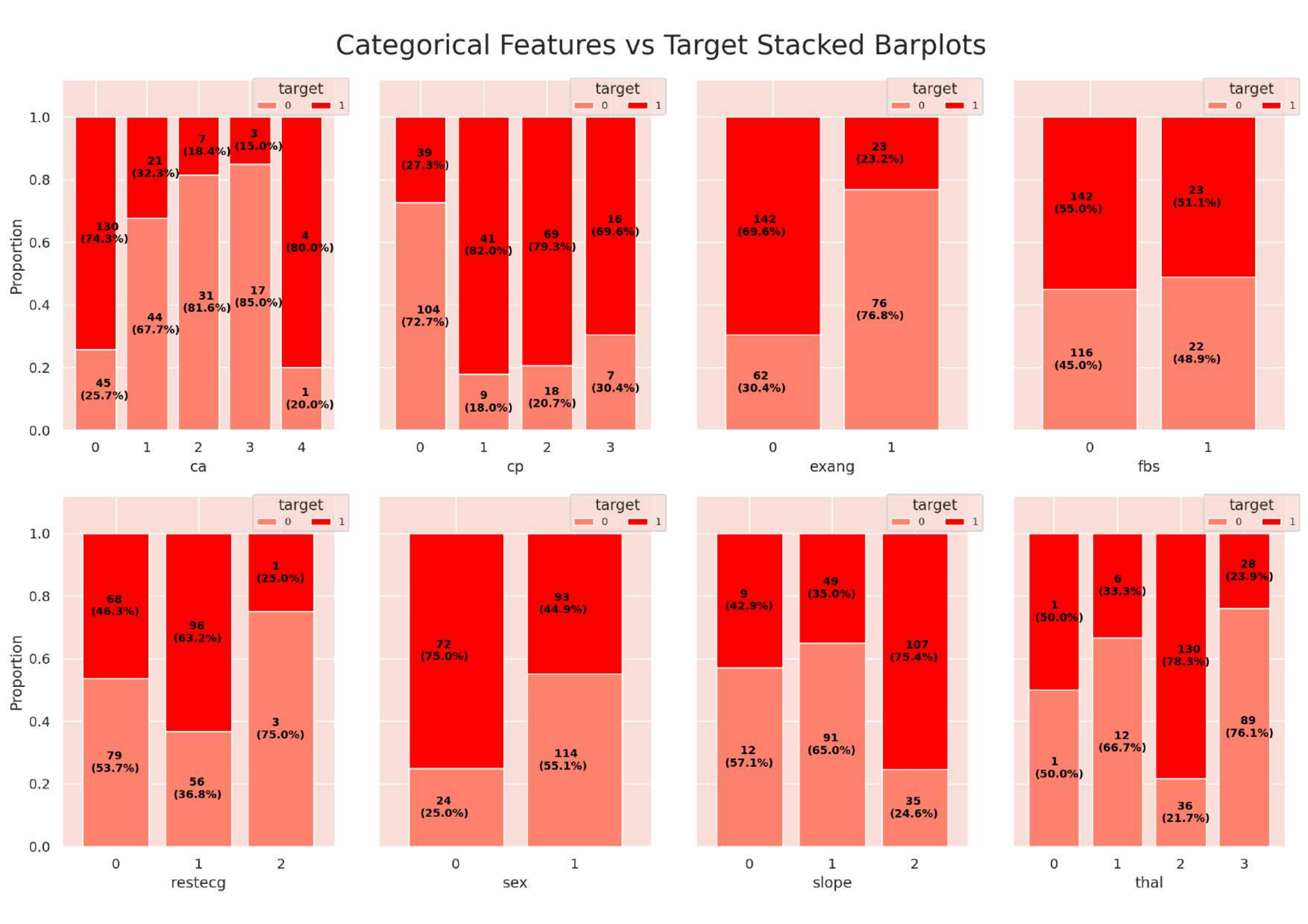
RAM Disk Colab AI

[19]
```
# Remove 'target' from the categorical_features
categorical_features = [feature for feature in categorical_features if feature != 'target']
```

```
fig, ax = plt.subplots(nrows=2, ncols=4, figsize=(15,10))

for i,col in enumerate(categorical_features):

    # Create a cross tabulation showing the proportion of purchased and non-purchased loans for each category of the feature
    cross_tab = pd.crosstab(index=df[col], columns=df['target'])

    # Using the normalize=True argument gives us the index-wise proportion of the data
    cross_tab_prop = pd.crosstab(index=df[col], columns=df['target'], normalize='index')

    # Define colormap
    cmp = ListedColormap(['#ff826e', 'red'])

    # Plot stacked bar charts
    x, y = i//4, i%4
    cross_tab_prop.plot(kind='bar', ax=ax[x,y], stacked=True, width=0.8, colormap=cmp,
                        legend=False, ylabel='Proportion', sharey=True)

    # Add the proportions and counts of the individual bars to our plot
    for idx, val in enumerate([*cross_tab.index.values]):
        for (proportion, count, y_location) in zip(cross_tab_prop.loc[val],cross_tab.loc[val],cross_tab_prop.loc[val].cumsum()):
            ax[x,y].text(x=idx-0.3, y=(y_location-proportion)+(proportion/2)-0.03,
                         s = f'    {count}\n({np.round(proportion * 100, 1)}%)',
                         color = "black", fontsize=9, fontweight="bold")

    # Add legend
    ax[x,y].legend(title='target', loc=(0.7,0.9), fontsize=8, ncol=2)
    # Set y limit
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:12 PM 04-04-2024


Categorical Features vs Target Stacked Barplots
target
0
1
Proportion
0.0
0.2
0.4
0.6
0.8
1.0
ca
0
1
2
3
4
130 (74.3%)
45 (25.7%)
21 (32.3%)
44 (67.7%)
7 (18.4%)
31 (81.6%)
3 (15.0%)
17 (85.0%)
4 (80.0%)
1 (20.0%)
cp
0
1
2
3
39 (27.3%)
104 (72.7%)
41 (82.0%)
9 (18.0%)
69 (79.3%)
18 (20.7%)
16 (69.6%)
7 (30.4%)
exang
0
1
142 (69.6%)
62 (30.4%)
23 (23.2%)
76 (76.8%)
fbs
0
1
142 (55.0%)
116 (45.0%)
23 (51.1%)
22 (48.9%)
restecg
0
1
2
68 (46.3%)
79 (53.7%)
96 (63.2%)
56 (36.8%)
1 (25.0%)
3 (75.0%)
sex
0
1
72 (75.0%)
24 (25.0%)
93 (44.9%)
114 (55.1%)
slope
0
1
2
9 (42.9%)
12 (57.1%)
49 (35.0%)
91 (65.0%)
107 (75.4%)
35 (24.6%)
thal
0
1
2
3
1 (50.0%)
1 (50.0%)
6 (33.3%)
12 (66.7%)
130 (78.3%)
36 (21.7%)
28 (23.9%)
89 (76.1%)

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNIeA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=k582hv9sRI01

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
[21] # Check for missing values in the dataset
     df.isnull().sum().sum()
```

```
0
```

```
[22] continuous_features
```

```
['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

```
Q1 = df[continuous_features].quantile(0.25)
Q3 = df[continuous_features].quantile(0.75)
IQR = Q3 - Q1
outliers_count_specified = ((df[continuous_features] < (Q1 - 1.5 * IQR)) | (df[continuous_features] > (Q3 + 1.5 * IQR))).sum()

outliers_count_specified
```

```
age         0
trestbps    9
chol        5
thalach     1
oldpeak     5
dtype: int64
```

```
[24] # Implementing one-hot encoding on the specified categorical features
     df_encoded = pd.get_dummies(df, columns=['cp', 'restecg', 'thal'], drop_first=True)

     # Convert the rest of the categorical variables that don't need one-hot encoding to integer data type
     features_to_convert = ['sex', 'fbs', 'exang', 'slope', 'ca', 'target']
     for feature in features_to_convert:
         df_encoded[feature] = df_encoded[feature].astype(int)
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:13 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=k582hv9sRI01

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

[24]
```
# Convert the rest of the categorical variables that don't need one-hot encoding to integer data type
features_to_convert = ['sex', 'fbs', 'exang', 'slope', 'ca', 'target']
for feature in features_to_convert:
    df_encoded[feature] = df_encoded[feature].astype(int)

df_encoded.dtypes
```

```
age           int64
sex           int64
trestbps      int64
chol          int64
fbs           int64
thalach       int64
exang         int64
oldpeak     float64
slope         int64
ca            int64
target        int64
cp_1           bool
cp_2           bool
cp_3           bool
restecg_1      bool
restecg_2      bool
thal_1         bool
thal_2         bool
thal_3         bool
dtype: object
```

[25]
```
# Displaying the resulting DataFrame after one-hot encoding
df_encoded.head()
```

	age	sex	trestbps	chol	fbs	thalach	exang	oldpeak	slope	ca	target	cp_1	cp_2	cp_3	restecg_1	restecg_2	thal_1	thal_2	thal_3

Connected to Python 3 Google Compute Engine backend

ENG IN 07:13 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNIeA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=k582hv9sRI01

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

[25]
```
# Displaying the resulting DataFrame after one-hot encoding
df_encoded.head()
```

	age	sex	trestbps	chol	fbs	thalach	exang	oldpeak	slope	ca	target	cp_1	cp_2	cp_3	restecg_1	restecg_2	thal_1	thal_2	thal_3
0	63	1	145	233	1	150	0	2.3	0	0	1	False	False	True	False	False	True	False	False
1	37	1	130	250	0	187	0	3.5	0	0	1	False	True	False	True	False	False	True	False
2	41	0	130	204	0	172	0	1.4	2	0	1	True	False	False	False	False	False	True	False
3	56	1	120	236	0	178	0	0.8	2	0	1	True	False	False	True	False	False	True	False
4	57	0	120	354	0	163	1	0.6	2	0	1	False	False	False	True	False	False	True	False

Next steps: Generate code with df_encoded View recommended plots

[26]
```
# Define the features (X) and the output labels (y)
X = df_encoded.drop('target', axis=1)
y = df_encoded['target']
```

[27]
```
# Splitting data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0, stratify=y)
```

[28]
```
continuous_features
```

```
['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

[29]
```
# Adding a small constant to 'oldpeak' to make all values positive
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:13 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=k582hv9sRI01

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

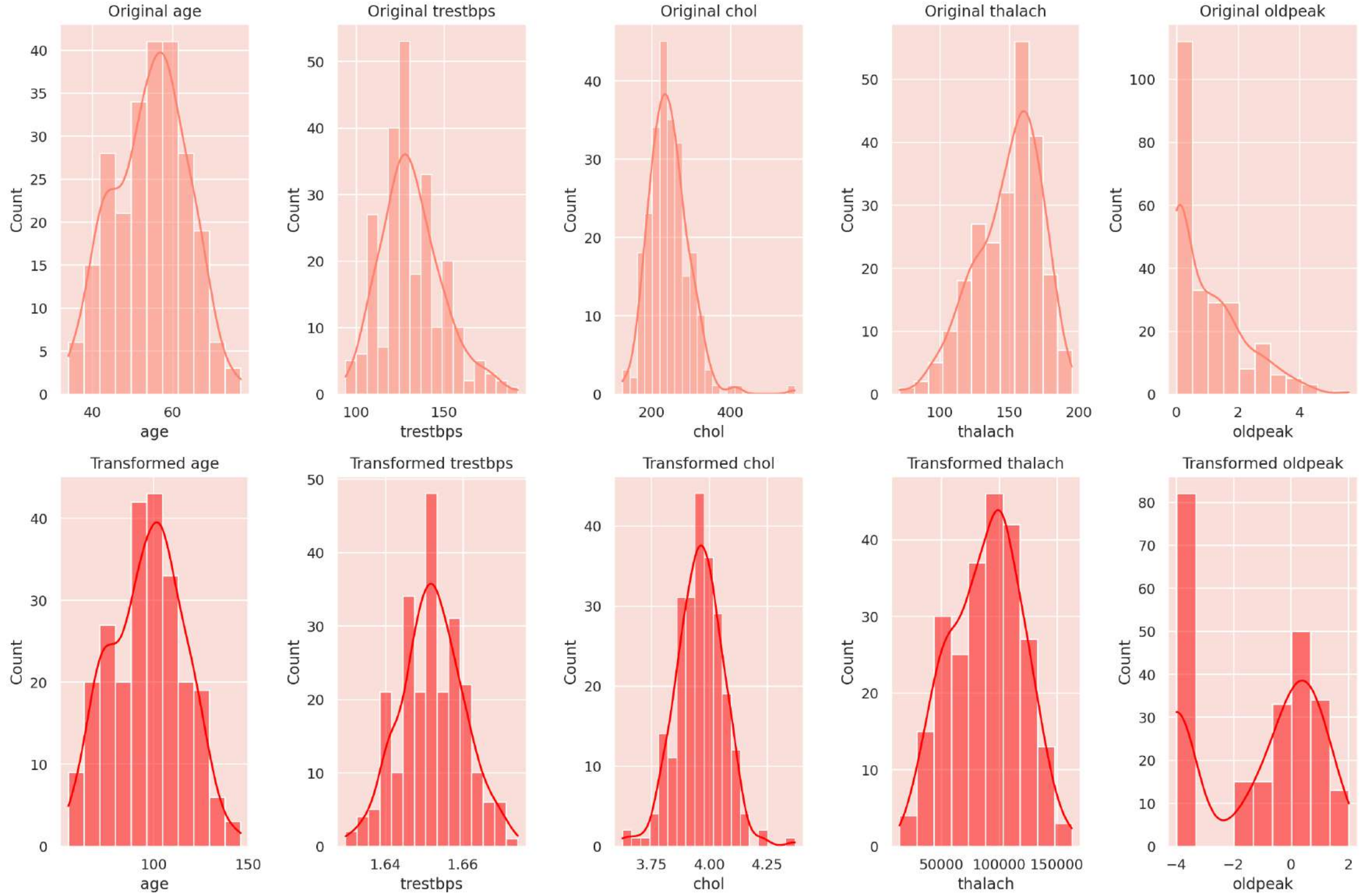
```
[29] # Adding a small constant to 'oldpeak' to make all values positive
     X_train['oldpeak'] = X_train['oldpeak'] + 0.001
     X_test['oldpeak'] = X_test['oldpeak'] + 0.001
```

```
[30] # Checking the distribution of the continuous features
     fig, ax = plt.subplots(2, 5, figsize=(15,10))

     # Original Distributions
     for i, col in enumerate(continuous_features):
         sns.histplot(X_train[col], kde=True, ax=ax[0,i], color='#ff826e').set_title(f'Original {col}')


     # Applying Box-Cox Transformation
     # Dictionary to store lambda values for each feature
     lambdas = {}

     for i, col in enumerate(continuous_features):
         # Only apply box-cox for positive values
         if X_train[col].min() > 0:
             X_train[col], lambdas[col] = boxcox(X_train[col])
             # Applying the same lambda to test data
             X_test[col] = boxcox(X_test[col], lmbda=lambdas[col])
             sns.histplot(X_train[col], kde=True, ax=ax[1,i], color='red').set_title(f'Transformed {col}')
         else:
             sns.histplot(X_train[col], kde=True, ax=ax[1,i], color='green').set_title(f'{col} (Not Transformed)')

     fig.tight_layout()
     plt.show()
```

Original age | Original trestbps | Original chol | Original thalach | Original oldpeak

Connected to Python 3 Google Compute Engine backend

ENG IN 07:14 PM 04-04-2024


Original age
Original trestbps
Original chol
Original thalach
Original oldpeak
Transformed age
Transformed trestbps
Transformed chol
Transformed thalach
Transformed oldpeak
Count
age
trestbps
chol
thalach
oldpeak

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNIeA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=nzGj5UzERhU9

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
X_train.head()
```

	age	sex	trestbps	chol	fbs	thalach	exang	oldpeak	slope	ca	cp_1	cp_2	cp_3	restecg_1	restecg_2	thal_1	thal_2	thal_3
269	99.775303	1	1.652121	4.044510	1	34193.175862	1	0.490856	0	0	False	False	False	False	False	False	False	True
191	104.060224	1	1.651136	3.909224	0	61564.541974	1	0.846853	1	3	False	False	False	False	False	False	False	True
15	87.096543	0	1.646937	3.916242	0	97354.732537	0	0.490856	1	0	False	True	False	True	False	False	True	False
224	95.519131	1	1.641028	3.960430	0	55975.802227	1	1.130195	1	1	False	False	False	True	False	False	False	True
250	89.190680	1	1.656716	4.069854	0	51729.405015	1	1.634849	1	3	False	False	False	True	False	False	False	True

Next steps: Generate code with X_train View recommended plots

```
[32] # Define the base DT model
     dt_base = DecisionTreeClassifier(random_state=0)
```

```
[33] def tune_clf_hyperparameters(clf, param_grid, X_train, y_train, scoring='recall', n_splits=3):
         '''
         This function optimizes the hyperparameters for a classifier by searching over a specified hyperparameter grid.
         It uses GridSearchCV and cross-validation (StratifiedKFold) to evaluate different combinations of hyperparameters.
         The combination with the highest recall for class 1 is selected as the default scoring metric.
         The function returns the classifier with the optimal hyperparameters.
         '''

         # Create the cross-validation object using StratifiedKFold to ensure the class distribution is the same across all the folds
         cv = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=0)
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:15 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNIeA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=nzGj5UzERhU9

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
[34] # Hyperparameter grid for DT
     param_grid_dt = {
         'criterion': ['gini', 'entropy'],
         'max_depth': [2,3],
         'min_samples_split': [2, 3, 4],
         'min_samples_leaf': [1, 2]
     }
```

```
[35] # Call the function for hyperparameter tuning
     best_dt, best_dt_hyperparams = tune_clf_hyperparameters(dt_base, param_grid_dt, X_train, y_train)
```

```
[36] print('DT Optimal Hyperparameters: \n', best_dt_hyperparams)
```

```
DT Optimal Hyperparameters:
 {'criterion': 'entropy', 'max_depth': 2, 'min_samples_leaf': 1, 'min_samples_split': 2}
```

```
[37] # Evaluate the optimized model on the train data
     print(classification_report(y_train, best_dt.predict(X_train)))
```

	precision	recall	f1-score	support
0	0.73	0.75	0.74	110
1	0.78	0.77	0.78	132
accuracy			0.76	242
macro avg	0.76	0.76	0.76	242
weighted avg	0.76	0.76	0.76	242

```
[38] # Evaluate the optimized model on the test data
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:15 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=nzGj5UzERhU9

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

[38]
```
# Evaluate the optimized model on the test data
print(classification_report(y_test, best_dt.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.80      0.71      0.75        28
           1       0.78      0.85      0.81        33

    accuracy                           0.79        61
   macro avg       0.79      0.78      0.78        61
weighted avg       0.79      0.79      0.79        61
```

[39]
```
def evaluate_model(model, X_test, y_test, model_name):
    """
    Evaluates the performance of a trained model on test data using various metrics.
    """
    # Make predictions
    y_pred = model.predict(X_test)

    # Get classification report
    report = classification_report(y_test, y_pred, output_dict=True)

    # Extracting metrics
    metrics = {
        "precision_0": report["0"]["precision"],
        "precision_1": report["1"]["precision"],
        "recall_0": report["0"]["recall"],
        "recall_1": report["1"]["recall"],
        "f1_0": report["0"]["f1-score"],
        "f1_1": report["1"]["f1-score"],
        "macro_avg_precision": report["macro avg"]["precision"]
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:15 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNIeA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=nzGj5UzERhU9

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
[40] dt_evaluation = evaluate_model(best_dt, X_test, y_test, 'DT')
     dt_evaluation
```

	precision_0	precision_1	recall_0	recall_1	f1_0	f1_1	macro_avg_precision	macro_avg_recall	macro_avg_f1	accuracy
DT	0.8	0.78	0.71	0.85	0.75	0.81	0.79	0.78	0.78	0.79

```
[41] rf_base = RandomForestClassifier(random_state=0)
```

```
[42] param_grid_rf = {
         'n_estimators': [10, 30, 50, 70, 100],
         'criterion': ['gini', 'entropy'],
         'max_depth': [2, 3, 4],
         'min_samples_split': [2, 3, 4, 5],
         'min_samples_leaf': [1, 2, 3],
         'bootstrap': [True, False]
     }
```

```
[43] # Using the tune_clf_hyperparameters function to get the best estimator
     best_rf, best_rf_hyperparams = tune_clf_hyperparameters(rf_base, param_grid_rf, X_train, y_train)
     print('RF Optimal Hyperparameters: \n', best_rf_hyperparams)
```

```
RF Optimal Hyperparameters:
 {'bootstrap': True, 'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 3, 'min_samples_split': 2, 'n_estimators': 30}
```

```
[44] # Evaluate the optimized model on the train data
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:15 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=nzGj5UzERhU9

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
{'bootstrap': True, 'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 3, 'min_samples_split': 2, 'n_estimators': 30}
```

[44]
```
# Evaluate the optimized model on the train data
print(classification_report(y_train, best_rf.predict(X_train)))
```

	precision	recall	f1-score	support
0	0.84	0.79	0.81	110
1	0.83	0.87	0.85	132
accuracy			0.83	242
macro avg	0.83	0.83	0.83	242
weighted avg	0.83	0.83	0.83	242

[45]
```
# Evaluate the optimized model on the test data
print(classification_report(y_test, best_rf.predict(X_test)))
```

	precision	recall	f1-score	support
0	0.85	0.79	0.81	28
1	0.83	0.88	0.85	33
accuracy			0.84	61
macro avg	0.84	0.83	0.83	61
weighted avg	0.84	0.84	0.84	61

[46]
```
rf_evaluation = evaluate_model(best_rf, X_test, y_test, 'RF')
rf_evaluation
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:15 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=nzGj5UzERhU9

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

[46] rf_evaluation = evaluate_model(best_rf, X_test, y_test, 'RF')
rf_evaluation

	precision_0	precision_1	recall_0	recall_1	f1_0	f1_1	macro_avg_precision	macro_avg_recall	macro_avg_f1	accuracy
RF	0.85	0.83	0.79	0.88	0.81	0.85	0.84	0.83	0.83	0.84

[47] # Define the base KNN model and set up the pipeline with scaling
knn_pipeline = Pipeline([
('scaler', StandardScaler()),
('knn', KNeighborsClassifier())
])

[48] # Hyperparameter grid for KNN
knn_param_grid = {
'knn__n_neighbors': list(range(1, 12)),
'knn__weights': ['uniform', 'distance'],
'knn__p': [1, 2] # 1: Manhattan distance, 2: Euclidean distance
}

[49] # Hyperparameter tuning for KNN
best_knn, best_knn_hyperparams = tune_clf_hyperparameters(knn_pipeline, knn_param_grid, X_train, y_train)
print('KNN Optimal Hyperparameters: \n', best_knn_hyperparams)

KNN Optimal Hyperparameters:
{'knn__n_neighbors': 9, 'knn__p': 1, 'knn__weights': 'uniform'}

Connected to Python 3 Google Compute Engine backend

ENG IN 07:15 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=nzGj5UzERhU9

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
[49] best_knn, best_knn_hyperparams = tune_clf_hyperparameters(knn_pipeline, knn_param_grid, X_train, y_train)
     print('KNN Optimal Hyperparameters: \n', best_knn_hyperparams)
```

```
KNN Optimal Hyperparameters:
 {'knn__n_neighbors': 9, 'knn__p': 1, 'knn__weights': 'uniform'}
```

```
[50] # Evaluate the optimized model on the train data
     print(classification_report(y_train, best_knn.predict(X_train)))
```

```
              precision    recall  f1-score   support

           0       0.80      0.79      0.79       110
           1       0.83      0.83      0.83       132

    accuracy                           0.81       242
   macro avg       0.81      0.81      0.81       242
weighted avg       0.81      0.81      0.81       242
```

```
[51] # Evaluate the optimized model on the test data
     print(classification_report(y_test, best_knn.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.82      0.82      0.82        28
           1       0.85      0.85      0.85        33

    accuracy                           0.84        61
   macro avg       0.83      0.83      0.83        61
weighted avg       0.84      0.84      0.84        61
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:16 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNIeA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=nzGj5UzERhU9

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

[52] knn_evaluation = evaluate_model(best_knn, X_test, y_test, 'KNN')
knn_evaluation

	precision_0	precision_1	recall_0	recall_1	f1_0	f1_1	macro_avg_precision	macro_avg_recall	macro_avg_f1	accuracy
KNN	0.82	0.85	0.82	0.85	0.82	0.85	0.83	0.83	0.83	0.84

[53] svm_pipeline = Pipeline([
('scaler', StandardScaler()),
('svm', SVC(probability=True))
])

[54] param_grid_svm = {
'svm__C': [0.0011, 0.005, 0.01, 0.05, 0.1, 1, 10, 20],
'svm__kernel': ['linear', 'rbf', 'poly'],
'svm__gamma': ['scale', 'auto', 0.1, 0.5, 1, 5],
'svm__degree': [2, 3, 4]
}

[55] # Call the function for hyperparameter tuning
best_svm, best_svm_hyperparams = tune_clf_hyperparameters(svm_pipeline, param_grid_svm, X_train, y_train)
print('SVM Optimal Hyperparameters: \n', best_svm_hyperparams)

```
SVM Optimal Hyperparameters:
 {'svm__C': 0.0011, 'svm__degree': 2, 'svm__gamma': 'scale', 'svm__kernel': 'linear'}
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:16 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=nzGj5UzERhU9

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
[56] # Evaluate the optimized model on the train data
     print(classification_report(y_train, best_svm.predict(X_train)))
```

```
              precision    recall  f1-score   support

           0       0.92      0.54      0.68       110
           1       0.71      0.96      0.82       132

    accuracy                           0.77       242
   macro avg       0.82      0.75      0.75       242
weighted avg       0.81      0.77      0.76       242
```

```
[57] # Evaluate the optimized model on the test data
     print(classification_report(y_test, best_svm.predict(X_test)))
```

```
              precision    recall  f1-score   support

           0       0.94      0.57      0.71        28
           1       0.73      0.97      0.83        33

    accuracy                           0.79        61
   macro avg       0.83      0.77      0.77        61
weighted avg       0.83      0.79      0.78        61
```

```
[58] svm_evaluation = evaluate_model(best_svm, X_test, y_test, 'SVM')
     svm_evaluation
```

	precision_0	precision_1	recall_0	recall_1	f1_0	f1_1	macro_avg_precision	macro_avg_recall	macro_avg_f1	accuracy
SVM	0.94	0.73	0.57	0.97	0.71	0.83	0.83	0.77	0.77	0.79

Connected to Python 3 Google Compute Engine backend

ENG IN 07:16 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=nzGj5UzERhU9

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

```
[59] # Concatenate the dataframes
     all_evaluations = [dt_evaluation, rf_evaluation, knn_evaluation, svm_evaluation]
     results = pd.concat(all_evaluations)

     # Sort by 'recall_1'
     results = results.sort_values(by='recall_1', ascending=False).round(2)
     results
```

	precision_0	precision_1	recall_0	recall_1	f1_0	f1_1	macro_avg_precision	macro_avg_recall	macro_avg_f1	accuracy
SVM	0.94	0.73	0.57	0.97	0.71	0.83	0.83	0.77	0.77	0.79
RF	0.85	0.83	0.79	0.88	0.81	0.85	0.84	0.83	0.83	0.84
DT	0.80	0.78	0.71	0.85	0.75	0.81	0.79	0.78	0.78	0.79
KNN	0.82	0.85	0.82	0.85	0.82	0.85	0.83	0.83	0.83	0.84

Next steps: Generate code with results View recommended plots

```
[60] # Sort values based on 'recall_1'
     results.sort_values(by='recall_1', ascending=True, inplace=True)
     recall_1_scores = results['recall_1']

     # Plot the horizontal bar chart
     fig, ax = plt.subplots(figsize=(12, 7), dpi=70)
     ax.barh(results.index, recall_1_scores, color='red')

     # Annotate the values and indexes
     for i, (value, name) in enumerate(zip(recall_1_scores, results.index)):
         ax.text(value + 0.01, i, f"{value:.2f}", ha='left', va='center', fontweight='bold', color='red', fontsize=15)
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:16 PM 04-04-2024

Heart Disease Prediction | Heart Disease Prediction - Cola | Heart Disease Prediction using

colab.research.google.com/drive/1WRUrs8PNleA2Bk9V8oTOQnt1hSfxF_-c#scrollTo=nzGj5UzERhU9

Heart Disease Prediction

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

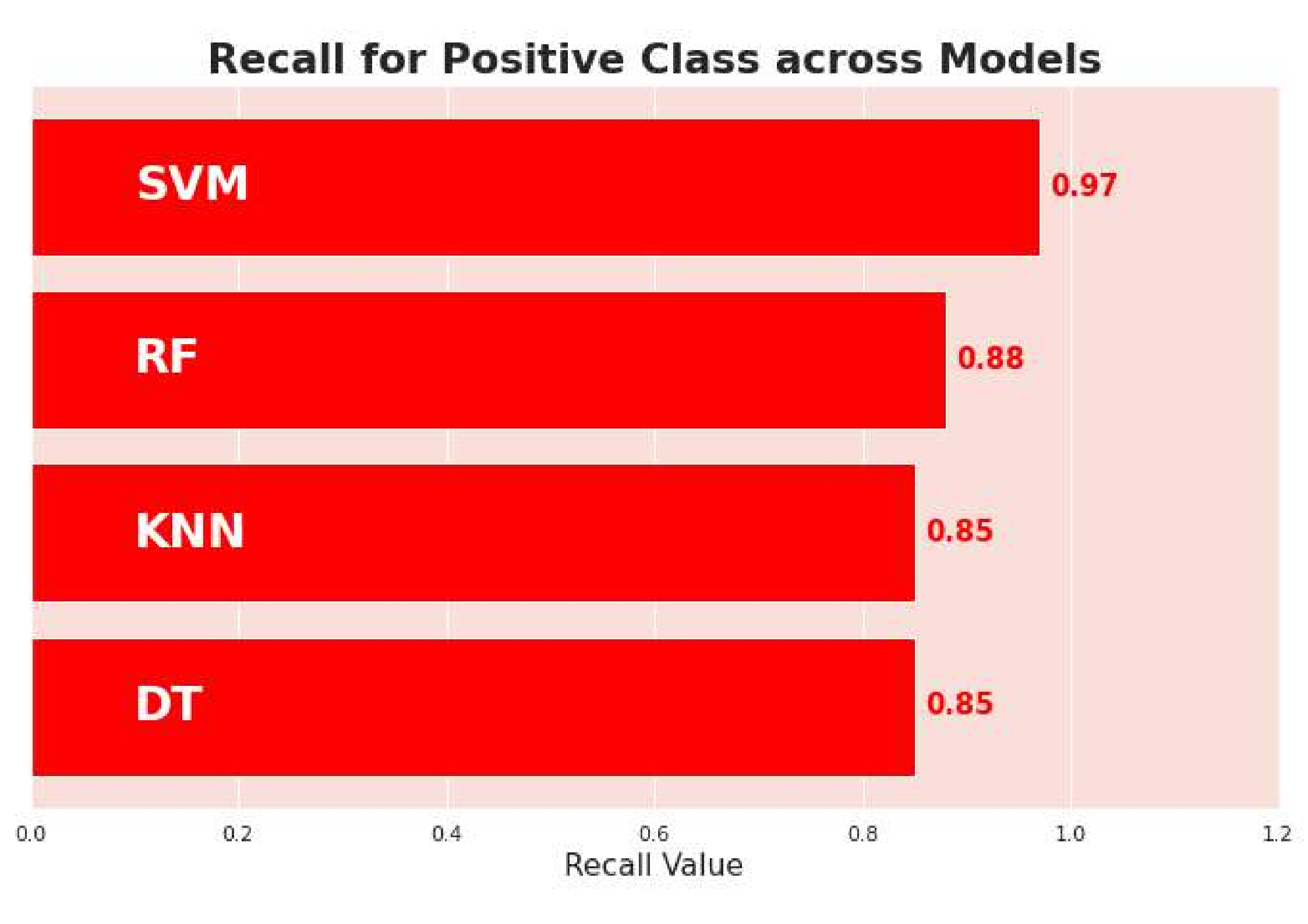
+ Code + Text

RAM Disk Colab AI

```
[60] # Sort values based on 'recall_1'
     results.sort_values(by='recall_1', ascending=True, inplace=True)
     recall_1_scores = results['recall_1']

     # Plot the horizontal bar chart
     fig, ax = plt.subplots(figsize=(12, 7), dpi=70)
     ax.barh(results.index, recall_1_scores, color='red')

     # Annotate the values and indexes
     for i, (value, name) in enumerate(zip(recall_1_scores, results.index)):
         ax.text(value + 0.01, i, f"{value:.2f}", ha='left', va='center', fontweight='bold', color='red', fontsize=15)
         ax.text(0.1, i, name, ha='left', va='center', fontweight='bold', color='white', fontsize=25)

     # Remove yticks
     ax.set_yticks([])

     # Set x-axis limit
     ax.set_xlim([0, 1.2])

     # Add title and xlabel
     plt.title("Recall for Positive Class across Models", fontweight='bold', fontsize=22)
     plt.xlabel('Recall Value', fontsize=16)
     plt.show()
```

Connected to Python 3 Google Compute Engine backend

ENG IN 07:16 PM 04-04-2024


Recall for Positive Class across Models
SVM
0.97
RF
0.88
KNN
0.85
DT
0.85
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Recall Value