STATS 3DA3

Homework Assignment 4

Lingyun Huang (400237999)

2024-03-13

Question 1

1. For this assignment, I will use the "heart disease" dataset available on Kaggle. Link to the dataset: Heart Disease Dataset on Kaggle
2. Load the dataset into a pandas DataFrame for analysis.

```
import pandas as pd
```

```
df = pd.read_csv('/Users/lingyunhuang/Desktop/3da_files/assignment4/framingham.csv')
df.head()
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes
0	1	39	4.0	0	0.0	0.0	0	0	0
1	0	46	2.0	0	0.0	0.0	0	0	0
2	1	48	1.0	1	20.0	0.0	0	0	0
3	0	61	3.0	1	30.0	0.0	0	1	0
4	0	46	3.0	1	23.0	0.0	0	0	0

3. Perform an initial exploration of the dataset to understand its structure (the number of features, observations, and variable types). Write at least three findings from the exploratory data analysis.

```
# describe the dataset
df.columns
```

```
Index(['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

```
df.shape
```

```
(4238, 16)
```

```
df.dtypes
```

```
male                int64
age                 int64
education         float64
currentSmoker       int64
cigsPerDay        float64
BPMeds            float64
prevalentStroke     int64
prevalentHyp        int64
diabetes            int64
totChol           float64
sysBP             float64
diaBP             float64
BMI               float64
heartRate         float64
glucose           float64
TenYearCHD          int64
dtype: object
```

There are 4238 observations (rows) in this dataset.

There are 16 features (columns) in this dataset.

Features in this dataset are 'male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds', 'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP', 'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'

Integer type features are `male`, `age`, `currentSmoker`, `prevalentStroke`, `prevalentHyp`, `diabetes`, `TenYearCHD`.

Float type features are `education`, `cigsPerDay`, `BPMeds`, `totChol`, `sysBP`, `diaBP`, `BMI`, `heartRate`, `glucose`.

4. Generate summary statistics for the dataset, including description on the categorical variables. Comment on the results (at least two statements).

```
df.describe()
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke
count	4238.000000	4238.000000	4133.000000	4238.000000	4209.000000	4185.000000	4238.000000
mean	0.429212	49.584946	1.978950	0.494101	9.003089	0.029630	0.005899
std	0.495022	8.572160	1.019791	0.500024	11.920094	0.169584	0.076587
min	0.000000	32.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	42.000000	1.000000	0.000000	0.000000	0.000000	0.000000
50%	0.000000	49.000000	2.000000	0.000000	0.000000	0.000000	0.000000
75%	1.000000	56.000000	3.000000	1.000000	20.000000	0.000000	0.000000
max	1.000000	70.000000	4.000000	1.000000	70.000000	1.000000	1.000000

Categorical variables in this dataset are `male`, `education`, `currentSmoker`, `BPMeds`, `prevalentStroke`, `prevalentHyp`, `diabetes`, `TenYearCHD`.

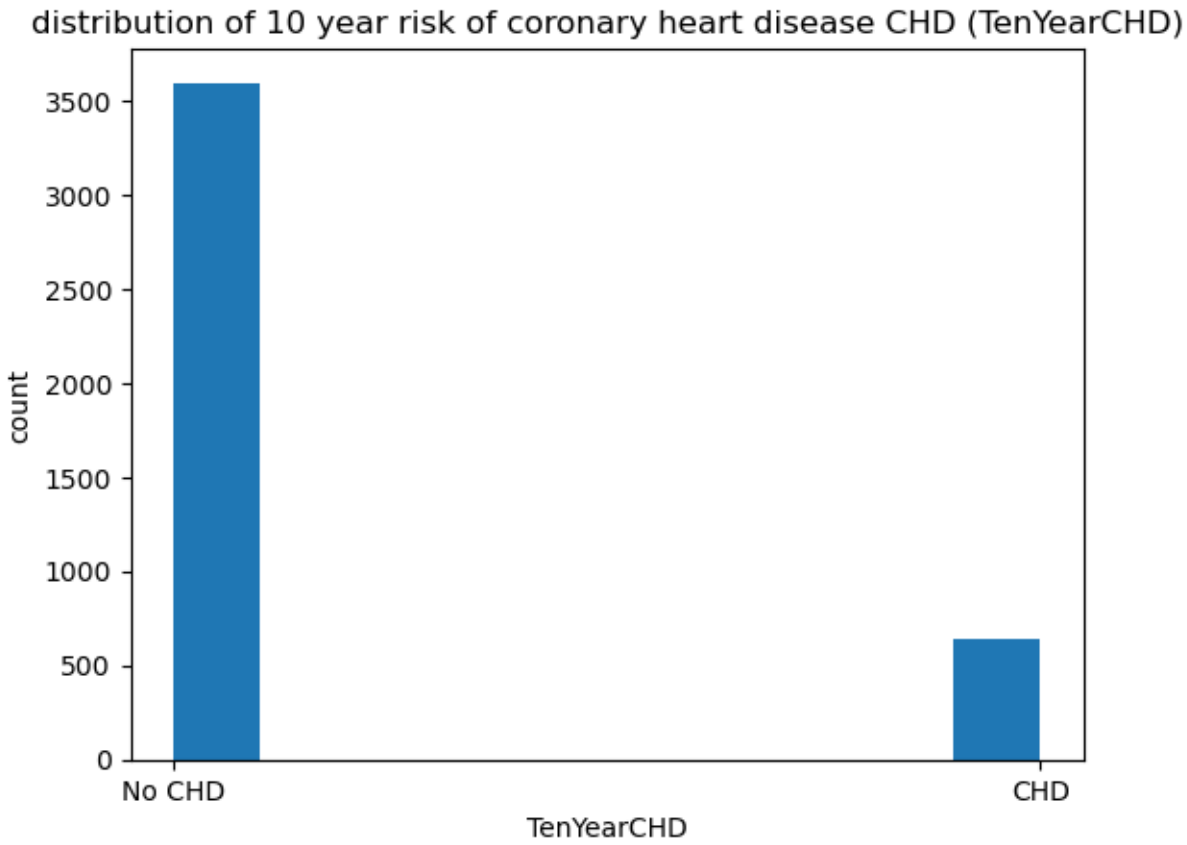
By reviewing count of each feature, some features like `age`, `education`, etc. may have missing values for their counts are lower than other features.

For some features like `glucose`, the maximum value is 394 mg/dL, which is notably higher than the 75% value of 87 mg/dL, outliers may exist.

5. Visualize the distribution of `10 year risk of coronary heart disease CHD (TenYearCHD)`. Comment on the plot (at least one statement).

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
plt.hist(df['TenYearCHD'])
plt.title('distribution of 10 year risk of coronary heart disease CHD (TenYearCHD)')
plt.xticks([0, 1], ['No CHD', 'CHD'])
plt.xlabel('TenYearCHD')
plt.ylabel('count')
plt.show()
```



The histogram shows that approximately 3600 individuals categorized as having no CHD (labeled as 0) and around 600 individuals categorized as having CHD (labeled as 1). Having no CHD (labeled as 0) is significantly more than having CHD (labeled as 1).

6. Are there any missing values? If you identify any, drop those observations with the missing values.

```
# check if missing values exist
missing_values = df.isnull().sum()
missing_values
```

```
male                0
age                 0
education         105
currentSmoker       0
cigsPerDay         29
```

```
BPMeds              53
prevalentStroke      0
prevalentHyp         0
diabetes             0
totChol             50
sysBP                0
diaBP                0
BMI                 19
heartRate            1
glucose            388
TenYearCHD           0
dtype: int64
```

```
# drop missing value
df_nona = df.dropna()


# check if all missing values are removed
print(df_nona.isnull().sum())
```

```
male               0
age                0
education          0
currentSmoker      0
cigsPerDay         0
BPMeds             0
prevalentStroke    0
prevalentHyp       0
diabetes           0
totChol            0
sysBP              0
diaBP              0
BMI                0
```

```
heartRate          0
glucose            0
TenYearCHD         0
dtype: int64
```

7. Skip the outlier analysis. Standardize the numerical predictor variables to ensure they are on the same scale.

```
from sklearn.preprocessing import StandardScaler
```

```
# Identify the numerical columns
df_num = df_nona.select_dtypes(include='float64').columns


# Scale the selected columns
scaler = StandardScaler()
df_nona[df_num] = scaler.fit_transform(df_nona[df_num])


# check if is scaled
df_nona.head()
```

```
/var/folders/jj/k27ndf3n34z0bf8kx6q4_fqc0000gn/T/ipykernel_870/1317636545.py:6: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/i
  df_nona[df_num] = scaler.fit_transform(df_nona[df_num])
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes
0	1	39	1.975752	0	-0.757068	-0.176951	0	0	0
1	0	46	0.019795	0	-0.757068	-0.176951	0	0	0
2	1	48	-0.958183	1	0.921174	-0.176951	0	0	0

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes
3	0	61	0.997773	1	1.760294	-0.176951	0	1	0
4	0	46	0.997773	1	1.172910	-0.176951	0	0	0

8. Split the dataset into a training set(75%) and a testing set (25%).

```
from sklearn.model_selection import train_test_split
```

```
x = df_nona.drop('TenYearCHD', axis = 1)
y = df_nona.TenYearCHD
```

```
x_train, x_test, y_train, y_test = train_test_split(
    x,
    y,
    test_size=0.25,
    random_state=0,
    stratify=y
    )
```

9. Create an instance of the logistic regression using `scikit-learn`.

```
from sklearn.linear_model import LogisticRegression
```

```
def_log = LogisticRegression()
```

10. Train the model on the training set and make predictions on the test set.

```
def_log.fit(x_train, y_train)
```

```
/Users/lingyunhuang/anaconda3/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:45
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
```

```
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

```
pred_prob = def_log.predict_proba(x_test)
pred_prob
```

```
array([[0.71030365, 0.28969635],
       [0.97715295, 0.02284705],
       [0.84272247, 0.15727753],
       ...,
       [0.90337918, 0.09662082],
       [0.95781101, 0.04218899],
       [0.79027244, 0.20972756]])
```

11. Calculate the accuracy of the model with the probability cut-off = 0.5 being classified into 10 year risk of coronary heart disease (CHD). Discuss the model's performance (compare the model's performance to flipping a coin)

```
# Create a data frame with predicted probabilities (for default) and the class labels in the te
df_logit = pd.DataFrame(
    data = {'prob0': pred_prob[:,1], 'y_test': y_test}
    )
```

```
df_logit.head()
```

	prob0	y_test
2215	0.289696	1
1309	0.022847	0

	prob0	y_test
863	0.157278	0
3256	0.088677	0
3235	0.059285	0

```
# Use cutoff = 0.5 and compute misclassification error, sensitivity, and specificity.
df_logit['y_test_pred'] = df_logit.prob0.map(lambda x: 1 if x>0.5 else 0)
```

```
df_logit.head()
```

	prob0	y_test	y_test_pred
2215	0.289696	1	0
1309	0.022847	0	0
863	0.157278	0	0
3256	0.088677	0	0
3235	0.059285	0	0

```
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, roc_auc_score
```

```
# Confusion Matrix
cm = confusion_matrix(df_logit.y_test, df_logit.y_test_pred)
print('Confusion Matrix : \n', cm)
```

```
Confusion Matrix :
 [[770   5]
 [130   9]]
```

```
# From confusion matrix calculate accuracy
total = sum(sum(cm))


accuracy = (cm[0,0]+cm[1,1])/total
```

```
print ('Accuracy : ', accuracy)


sensitivity = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity )


specificity = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity)
```

```
Accuracy :  0.8522975929978118
Sensitivity :  0.9935483870967742
Specificity :  0.06474820143884892
```

Accuracy of the model is around 0.85. The model's performance is really good, and much better than flipping a coin.

12. Comment on the sensitivity and specificity of the model (at least one statement).

```
print(classification_report(df_logit.y_test, df_logit.y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.99      0.92       775
           1       0.64      0.06      0.12       139

    accuracy                           0.85       914
   macro avg       0.75      0.53      0.52       914
weighted avg       0.82      0.85      0.80       914
```

The sensitivity is the proportion of diseased cases correctly predicted as such (true positives). The sensitivity is very high at approximately 99.35%, suggesting that the model performs exceptionally well in correctly identifying individuals with CHD among all actual positive cases.
The specificity is the proportion of non-diseased cases correctly predicted as such (true negatives).

The specificity is significantly lower at around 6.47%, indicating that the model has a high rate of false positives, incorrectly classifying individuals without CHD as having CHD.

13. Discuss any potential improvements based on the accuracy and confusion matrix.

Based on the accuracy and confusion matrix, we may need to adjust the model by using different cut-off values. Additionally, increasing the sample size could help improve model fitting.

14. Perform ROC analysis and plot the ROC curve. Additionally, calculate the area under the ROC curve (AUC). Determine the optimal probability cut-off point.

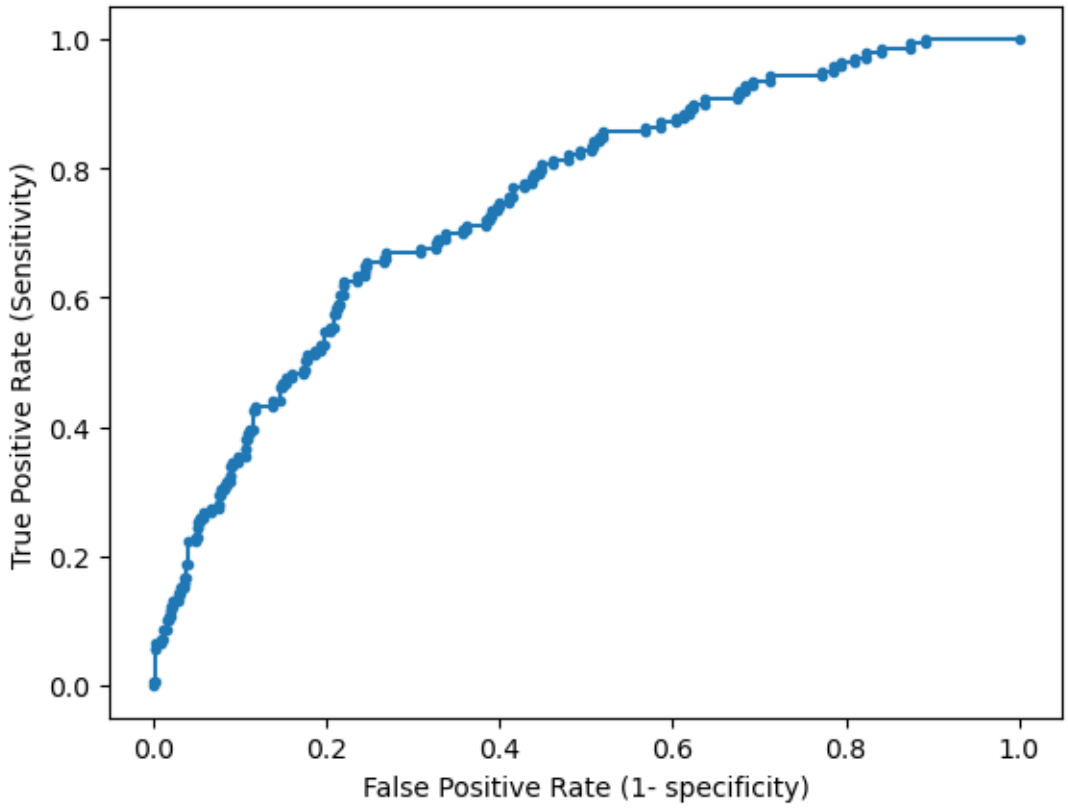
```
fpr, tpr, thresholds = roc_curve(df_logit.y_test, df_logit.prob0)
```

```
# AUC
roc_auc_score(df_logit.y_test, df_logit.prob0)
```

```
0.7490833139939661
```

```
# plot the roc curve for the model
plt.plot(fpr, tpr, marker='.', label='Logistic')
plt.xlabel('False Positive Rate (1- specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')

plt.show()
```



ROC Analysis of Binary Classifier (Youden's J statistic)

```
j_statistic = tpr - fpr
optimal_index = np.argmax(j_statistic)
optimal_threshold = thresholds[optimal_index]
optimal_threshold
```

```
0.170087271471232
```

```
ind = np.where(np.isclose(thresholds, optimal_threshold, atol=0.001))
print(tpr[ind])
print(1-fpr[ind])
```

```
[0.64028777 0.64748201 0.64748201 0.65467626]
[0.75483871 0.75483871 0.75354839 0.75354839]
```

ROC Analysis of Binary Classifier (Kolmogorov–Smirnov statistic)

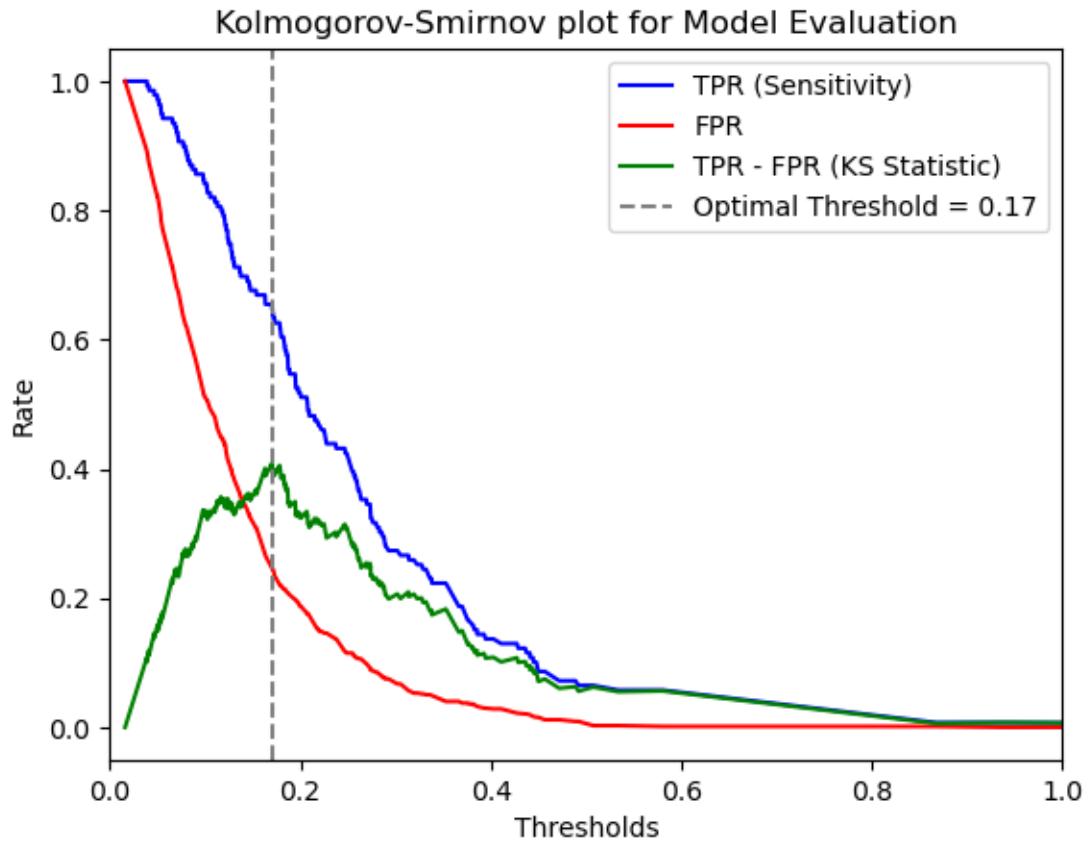
```
ks_statistic = np.max(tpr - fpr)
ks_threshold = thresholds[np.argmax(tpr - fpr)]
ks_threshold
```

```
0.170087271471232
```

```
ind = np.where(np.isclose(thresholds, ks_threshold, atol=0.001))
print(tpr[ind])
print(1-fpr[ind])
```

```
[0.64028777 0.64748201 0.64748201 0.65467626]
[0.75483871 0.75483871 0.75354839 0.75354839]
```

```
plt.plot(thresholds, tpr, label='TPR (Sensitivity)', color='blue')
plt.plot(thresholds, fpr, label='FPR', color='red')
plt.plot(thresholds, tpr - fpr, label='TPR - FPR (KS Statistic)', color='green')
plt.axvline(x=ks_threshold, color='grey', linestyle='--', label=f'Optimal Threshold = {ks_thres
plt.title('Kolmogorov-Smirnov plot for Model Evaluation')
plt.xlabel('Thresholds')
plt.ylabel('Rate')
plt.legend()
plt.xlim([0.0, 1.0])
#plt.gca().invert_xaxis()
plt.show()
```



The area under the ROC curve (AUC) is around 0.75.

From the result of Youden's J statistics and Kolmogorov-Smirnov statistics, the optimal probability cut-off point is at around 0.17.

15. The remaining questions will use the selected predictor variables: 'male', 'age', 'cigsPerDay', 'diabetes', 'sysBP', 'heartRate'. Create a design matrix with thsese predictors.

```
from patsy import dmatrices, dmatrix
```

```
X = dmatrix(
    'male + age + cigsPerDay + diabetes + sysBP + heartRate',
    data=df_nona,
    return_type='dataframe'
    )
X.head()
```

	Intercept	male	age	cigsPerDay	diabetes	sysBP	heartRate
0	1.0	1.0	39.0	-0.757068	0.0	-1.193695	0.356340
1	1.0	0.0	46.0	-0.757068	0.0	-0.514637	1.608289
2	1.0	1.0	48.0	0.921174	0.0	-0.220378	-0.060977
3	1.0	0.0	61.0	1.760294	0.0	0.798209	-0.895610
4	1.0	0.0	46.0	1.172910	0.0	-0.107202	0.773656

16. What are the significant variables in predicting the 10-year risk of coronary heart disease ('TenYearCHD') and why?

```
import statsmodels.api as sm
```

```
# split the data train/test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=0, stratify=y)
```

```
model_new = sm.Logit(y_train, X_train).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.383478
         Iterations 7
```

```
model_new.summary()
```

Dep. Variable:	TenYearCHD	**No. Observations:**	2742
Model:	Logit	**Df Residuals:**	2735
Method:	MLE	**Df Model:**	6
Date:	Wed, 13 Mar 2024	**Pseudo R-squ.:**	0.1018
Time:	12:03:30	**Log-Likelihood:**	-1051.5
converged:	True	**LL-Null:**	-1170.6
Covariance Type:	nonrobust	**LLR p-value:**	1.322e-48

	coef	std err	z	P> |z|	[0.025	0.975]
Intercept	-5.2075	0.394	-13.210	0.000	-5.980	-4.435
male	0.5100	0.123	4.157	0.000	0.270	0.750
age	0.0607	0.007	8.236	0.000	0.046	0.075
cigsPerDay	0.2208	0.057	3.884	0.000	0.109	0.332
diabetes	0.5652	0.273	2.067	0.039	0.029	1.101
sysBP	0.4247	0.055	7.737	0.000	0.317	0.532
heartRate	0.0054	0.056	0.097	0.923	-0.104	0.115

From the Logit Regression Results, p-value of `heartRate` is 0.923, which is greater than alpha = 0.05, hence the coefficient for `heartRate` is not statistically significant at the 0.05 significance level (p-value > 0.05), `heartRate` may not be a significant predictor of the 10-year risk of coronary heart disease in this model.

Therefore significant variables in predicting the 10-year risk of coronary heart disease (`TenYearCHD`) are `male`, `age`, `cigsPerDay`, `diabetes`, `sysBP`. Because their coefficients are statistically significant at the 0.05 significance level (p-value < 0.05), they are significant predictors of the 10-year risk of coronary heart disease in this model.

17. Interpret the coefficient of diabetes in predicting the 10 year risk of coronary heart disease ('TenYearCHD')

The coefficient of diabetes in predicting the 10-year risk of coronary heart disease ('TenYearCHD') is 0.5652 with p-value = 0.039, statistical significance at the 0.05 significance level.

The positive coefficient of diabetes indicating that individuals with diabetes have a higher predicted probability of developing coronary heart disease over the next 10 years compared to those without diabetes.

For each unit increase in the presence of diabetes, the log-odds of developing coronary heart disease increases by 0.5652 units.

3. Helper's name.

N/A