Project 1

This dataset is adapted from the World Health Organization on Strokes (it's based on real data but is NOT REAL). Use this dataset to answer the following questions and perform the following tasks. Feel free to add extra cells as needed, but **clearly identify where each question is answered, both the code and Markdown cells**. Please remove any superflous code. Please put any written/typed responses in MARKDOWN CELLS.

Data Information

- `reg_to_vote` : 0 if no, 1 if yes.
- `age` : age of the patient in years.
- `hypertension` : 0 if the patient doesn't have hypertension, 1 if the patient has hypertension.
- `heart_disease` : 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease.
- `ever_married` : 0 if no, 1 if yes.
- `Residence_type` : 0 for Rural, 1 for Urban.
- `avg_glucose_level` : average glucose level in blood.
- `bmi` : body mass index.
- `smoking_status_smokes`, `smoking_status_formerly` : Whether or not the person smokes, or formerly smoked. If a person has 0's for both these columns, they never smoked.
- `stroke` : 1 if the patient had a stroke or 0 if not.
- `dog_owner` : 0 if no, 1 if yes.
- `income_in_k` : income in thousands
- `er_visits` : number of recorded Emergency Room visits in lifetime.
- `raccoons_to_fight` : number of racoons the patient belives they could fight off at once.
- `fast_food_budget_month` : amount (in US dollars) spent on fast food per month.

Part I: Logistic Regression

Build a logistic regression model to predict whether or not someone had a `stroke` based on **all** the other variables in the dataset.

1. Count the missing data per column, and remove rows with missing data (if any).

2. Use 10 fold cross validation for your model validation. Z-score your continuous/interval variables only. Store both the train and test accuracies to check for overfitting. **Is the model overfit? How can you tell?**
3. After completing steps 1-2, fit another logistic regression model on ALL of the data (no model validation; but do z score) using the same predictors as before, and put the coefficients into a dataframe called `coef`.
4. print out a confusion matrix for the model you made in part 3. **What does this confusion matrix tell you about your model? How can you tell?**

Part II: Data Exploration

The WHO has asked the following five questions, create **at least 1 ggplot graph per question** (using the above data + model when needed) to help answer each question, and **explicitly answer the question in a Markdown cell** below your graph. You may use other calculations to help support your answer but MUST pair it with a graph. Write your answer as if you were explaining it to a non-data scientist. You will be graded on the effectiveness and clarity of your graph, as well as the completeness, clarity, and correctness of your responses and justifications.

1. In this specific data set, do dog-owners over 50 have a higher average probability of stoke than non-dog owners who currently smoke? How can you tell? (Do not use the model for this question, it's asking you to compare the observed probability of having a stroke in the two groups described).
2. What is the relationship between average blood glucose and BMI? Is the relationship between those two variables different for people who are and are not registered to vote? How can you tell?
3. Is your logistic regression model most accurate for people who make less than 30k, between 30-90k, or over 90k? Discuss the potential accuracy *and* ethical implications if your model *were* more accurate for different groups (you can use the full model from part I-3 to check accuracy for each of these groups; DO NOT create/fit new models for each income range, use the model from part I-3 to calculate the accuracy for each of these groups.)
4. Which of the following variables is the strongest predictor of having a stroke (owning a dog, residence type, marriage, being registered to vote)? How were you able to tell?
5. Create a variable `er_visits_per_year` that calculates the # of visits to the ER that a person has had per year of life. Store this variable in your data frame (no need to include

this variable in the previous logistic regression model). Is the # of ER visits per year different for stroke and non-stroke patients? How can you tell?

```
# import necessary packages
import warnings
warnings.filterwarnings('ignore')
```

```
import pandas as pd
import numpy as np
from plotnine import *

from sklearn.linear_model import LinearRegression # Linear Regression Model
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score #mode
from sklearn.model_selection import train_test_split # simple TT split cv
from sklearn.metrics import mean_absolute_error as mae
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score
from sklearn.linear_model import LogisticRegression # Logisitc Regression Model
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.model_selection import KFold # k-fold cv
from sklearn.model_selection import LeaveOneOut #LOO cv
from sklearn.model_selection import cross_val_score # cross validation metrics
from sklearn.model_selection import cross_val_predict # cross validation metrics
from sklearn.metrics import confusion_matrix

import time
%matplotlib inline
```

Part I: Logistic Regression

Build a logistic regression model to predict whether or not someone had a stroke based on all the other variables in the dataset.

```
DF = pd.read_csv("https://raw.githubusercontent.com/cmparlettpelleriti/CPSC392Parle
```

1. Count the missing data per column, and remove rows with missing data (if any).

```
#missing data stored in variable 'null'
#prints count of how many missing data values in each column
nullCount = DF.isna().sum()
print(nullCount)
```

```
age                          13
hypertension                 12
heart_disease                21
ever_married                  9
Residence_type               21
avg_glucose_level            31
bmi                         575
stroke                        0
smoking_status_smokes         0
smoking_status_formerly       0
reg_to_vote                  14
dog_owner                    21
raccoons_to_fight            27
fast_food_budget_month        8
income_in_k                  21
er_visits                    15
dtype: int64
```

```
#drops all rows with any NA values
DF.dropna(inplace = True)
print(DF)
```

```
           age  hypertension  heart_disease  ever_married  Residence_type  \
0      60.0           1.0            0.0           0.0             1.0
1       4.0           0.0            0.0           0.0             0.0
2      77.0           0.0            0.0           1.0             1.0
3      37.0           0.0            0.0           1.0             1.0
4      44.0           0.0            0.0           0.0             0.0
...     ...           ...            ...           ...             ...
14995  52.0           1.0            0.0           0.0             1.0
14996  56.0           0.0            0.0           1.0             1.0
14997  60.0           1.0            0.0           1.0             0.0
14998  77.0           0.0            0.0           1.0             0.0
14999  44.0           0.0            0.0           1.0             0.0

       avg_glucose_level   bmi  stroke  smoking_status_smokes  \
0                  73.00  25.2       0                      1
1                 110.15  17.1       0                      0
2                  68.38  27.8       0                      0
```

```
3                 95.08  30.1       0                    0
4                103.78  40.9       0                    1
...                 ...   ...     ...                  ...
14995            106.22  29.0       0                    0
14996             63.18  26.5       0                    0
14997            100.20  28.6       0                    0
14998             90.00  32.0       0                    0
14999             74.91  26.9       0                    0

       smoking_status_formerly  reg_to_vote  dog_owner  raccoons_to_fight  \
0                            0          1.0        1.0               10.0
1                            0          0.0        1.0               13.0
2                            0          0.0        1.0                6.0
3                            0          1.0        1.0               12.0
4                            0          1.0        1.0               11.0
...                        ...          ...        ...                ...
14995                        0          0.0        0.0                1.0
14996                        0          1.0        0.0               15.0
14997                        0          0.0        1.0               17.0
14998                        0          0.0        0.0               11.0
14999                        0          0.0        0.0                8.0

       fast_food_budget_month  income_in_k  er_visits
0                      209.19    51.553645        9.0
1                      176.46    45.405414        5.0
2                      213.00    94.865174        8.0
3                      161.90    84.123775        8.0
4                      261.29    74.794596       11.0
...                       ...          ...        ...
14995                  179.77    74.826197        9.0
14996                  143.61    52.280949       16.0
14997                   64.87    92.427118       24.0
14998                  205.92    53.042139        6.0
14999                  184.14    70.317707        4.0

[14222 rows x 16 columns]
```

2. Use 10 fold cross validation for your model validation. Z-score your continuous/interval variables only. Store both the train and test accuracies to check for overfitting. Is the model overfit? How can you tell?

```
k = KFold(n_splits = 10)
continuous_predictors = ["age", "avg_glucose_level", "bmi","raccoons_to_fight", "fa
y = DF['stroke']
X = DF[continuous_predictors]

#check number of splits
k.get_n_splits(X)

#build model
model = LogisticRegression()

#use for loop to iterate through folds and get training/testing accuracy
training_accuracy = []
testing_accuracy = []
for trainindex, testindex in k.split(X):
  X_train, X_test = X.iloc[trainindex], X.iloc[testindex]
  y_train, y_test = y.iloc[trainindex], y.iloc[testindex]
  model.fit(X_train, y_train)
  training_accuracy.append(model.score(X_train, y_train))
  testing_accuracy.append(model.score(X_test, y_test))

print(training_accuracy, testing_accuracy)
```

```
[0.9600750058598329, 0.9614813657316977, 0.960859375, 0.959296875, 0.96054687
```

The model is not overfit, because if it was, the training accuracy would be much higher than the testing accuracy, but that is not the case with this model.

#3) After completing steps 1-2, fit another logistic regression model on ALL of the data (no model validation; but do z score) using the same predictors as before, and put the coefficients into a dataframe called coef.

```
#logisitic regression on all of data
model2 = LogisticRegression()
model2.fit(X, y)
coef = pd.DataFrame(model2.coef_)

print(coef)
```

```
          0         1         2         3        4         5         6
0  0.073566  0.005923  0.015199 -0.003642  0.01738  0.012445  0.005354
```

4. print out a confusion matrix for the model you made in part 3. What does this confusion matrix tell you about your model? How can you tell?

```
#build confusion matrix
y_predict = model2.predict(X)
confusion_matrix(y, y_predict)
```

```
array([[13639,     7],
       [  557,    19]])
```

This confusion matrix shows in the top right shows who the model thought had a stroke, but didn't have a stroke, the top left shows who didn't have a stroke and actually didn't have a stroke, the bottom left shows who had a stroke but the model thought they didn't, and the bottom right shows the correctly guessed people who actually had strokes. The model predicts that most of the samples do not have a stroke because a majority of people in the data do not actually have strokes. But the model does not accurately guess people who did have strokes, as it missed 557 people who actually had a stroke. The model has high true negative rates and low true positive rates.

Part II: Data Exploration

The WHO has asked the following five questions, create at least 1 ggplot graph per question (using the above data + model when needed) to help answer each question, and explicitly answer the question in a Markdown cell below your graph. You may use other calculations to help support your answer but MUST pair it with a graph. Write your answer as if you were explaining it to a non-data scientist. You will be graded on the effectiveness and clarity of your graph, as well as the completeness, clarity, and correctness of your responses and justifications.

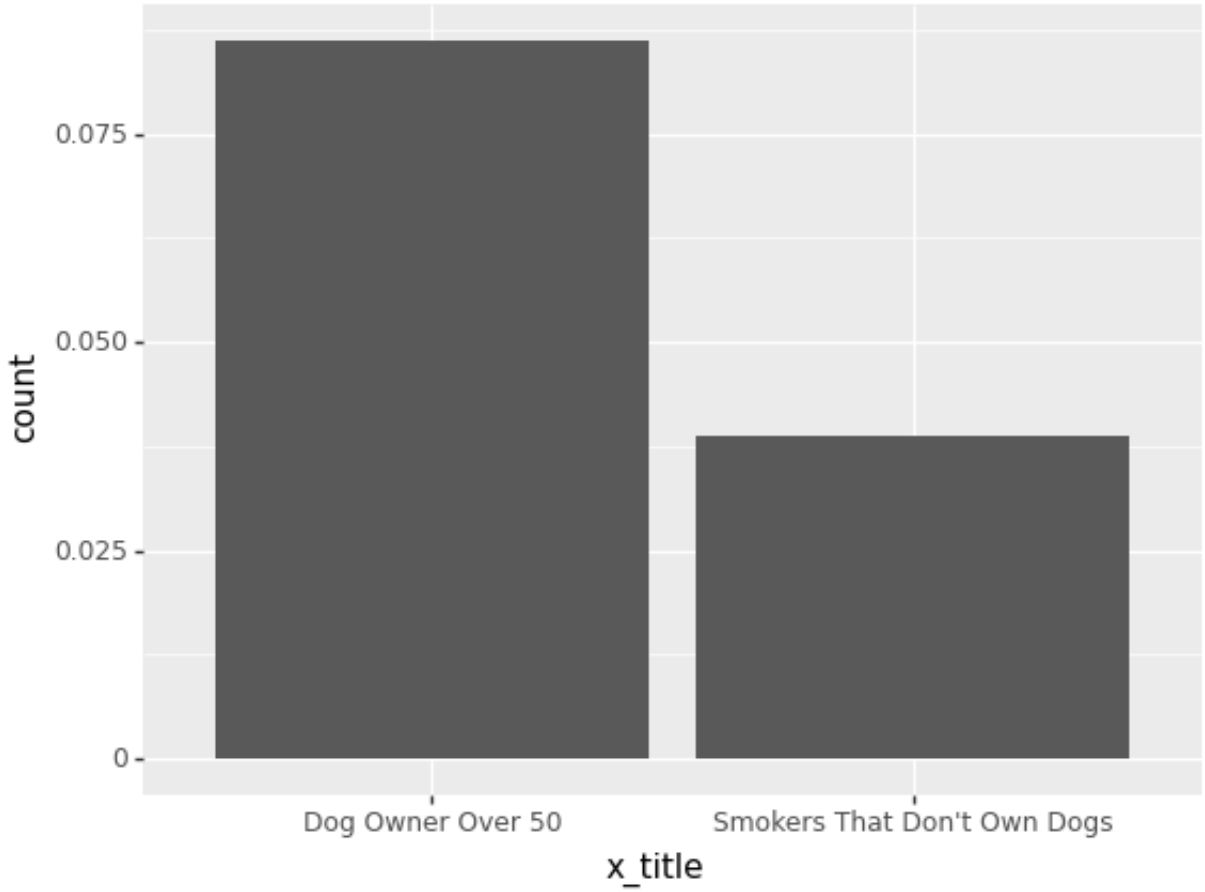
1. In this specific data set, do dog-owners over 50 have a higher average probability of stoke than non-dog owners who currently smoke? How can you tell? (Do not use the model for this question, it's asking you to compare the observed probability of having a stroke in the two groups described).

```
age_over_50 = DF[DF["age"] > 50]
dog_owner_over_50 = age_over_50[age_over_50["dog_owner"] == 1.0]
#print(dog_owner)

dog_prob = np.sum(dog_owner_over_50["stroke"]) / len(dog_owner_over_50["stroke"])


smokers = DF[DF["smoking_status_smokes"] == 1]
smoking_nondog_owner = smokers[smokers["dog_owner"] == 0]
#print(smoking_nondog_owner)
nondog_prob = np.sum(smoking_nondog_owner["stroke"]) / len(smoking_nondog_owner["st
```

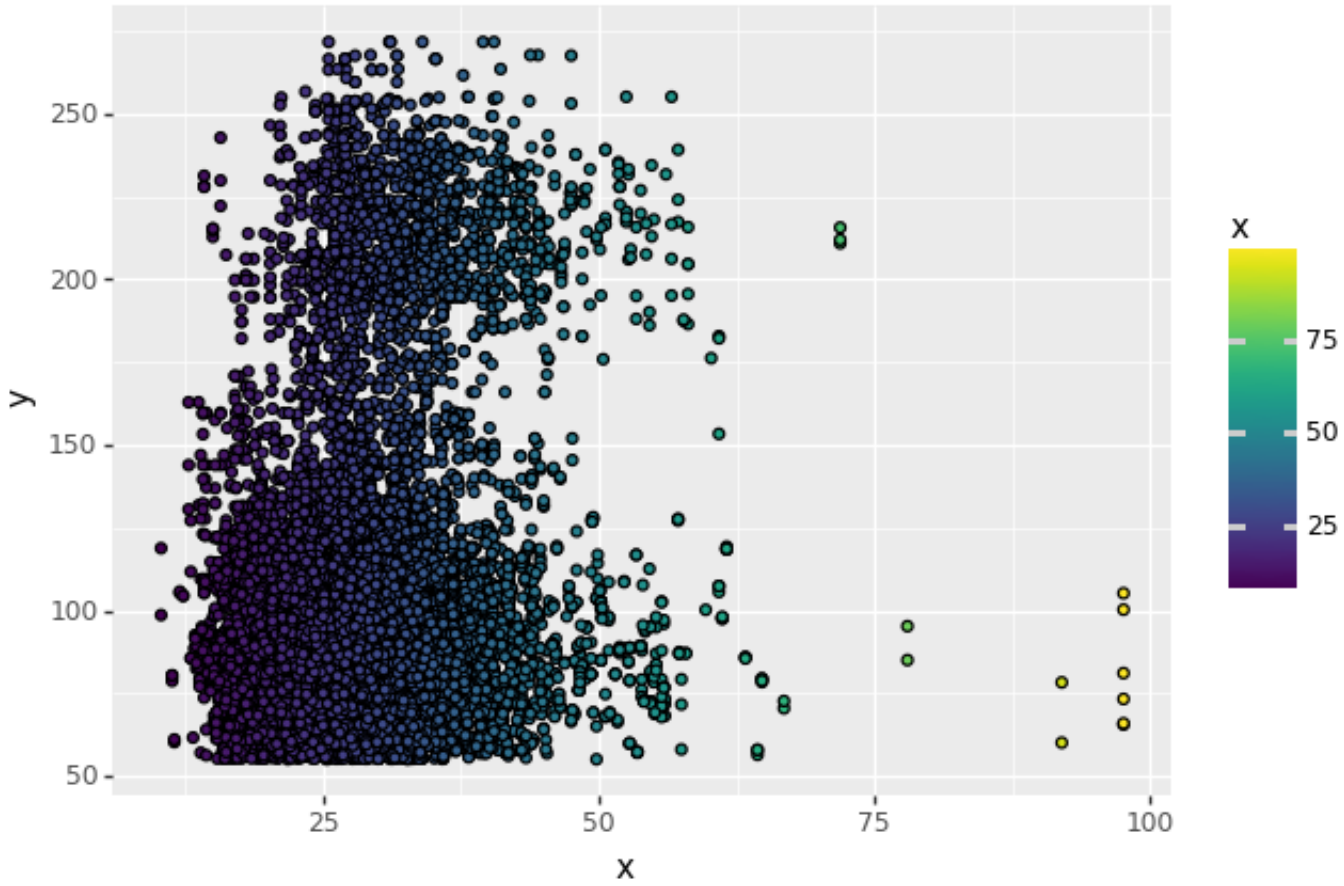
```
x_title = ["Dog Owner Over 50", "Smokers That Don't Own Dogs"]
y = [dog_prob, nondog_prob]
#df = pd.DataFrame({"x":[1,2,3,4], "y":[1,3,4,2]})
DF2 = pd.DataFrame({"y": y, "x_title": x_title})
DF2
ggplot(aes(x="x_title", weight="y"), DF2) + geom_bar()
```




```
<ggplot: (8775093937913)>
```

2. What is the relationship between average blood glucose and BMI? Is the relationship between those two variables different for people who are and are not registered to vote? How can you tell?

```
DF3 = pd.DataFrame({"x": DF["bmi"], "y": DF["avg_glucose_level"]})
DF3
ggplot(DF3, aes(x="x", y="y", fill = "x")) + geom_point()
```

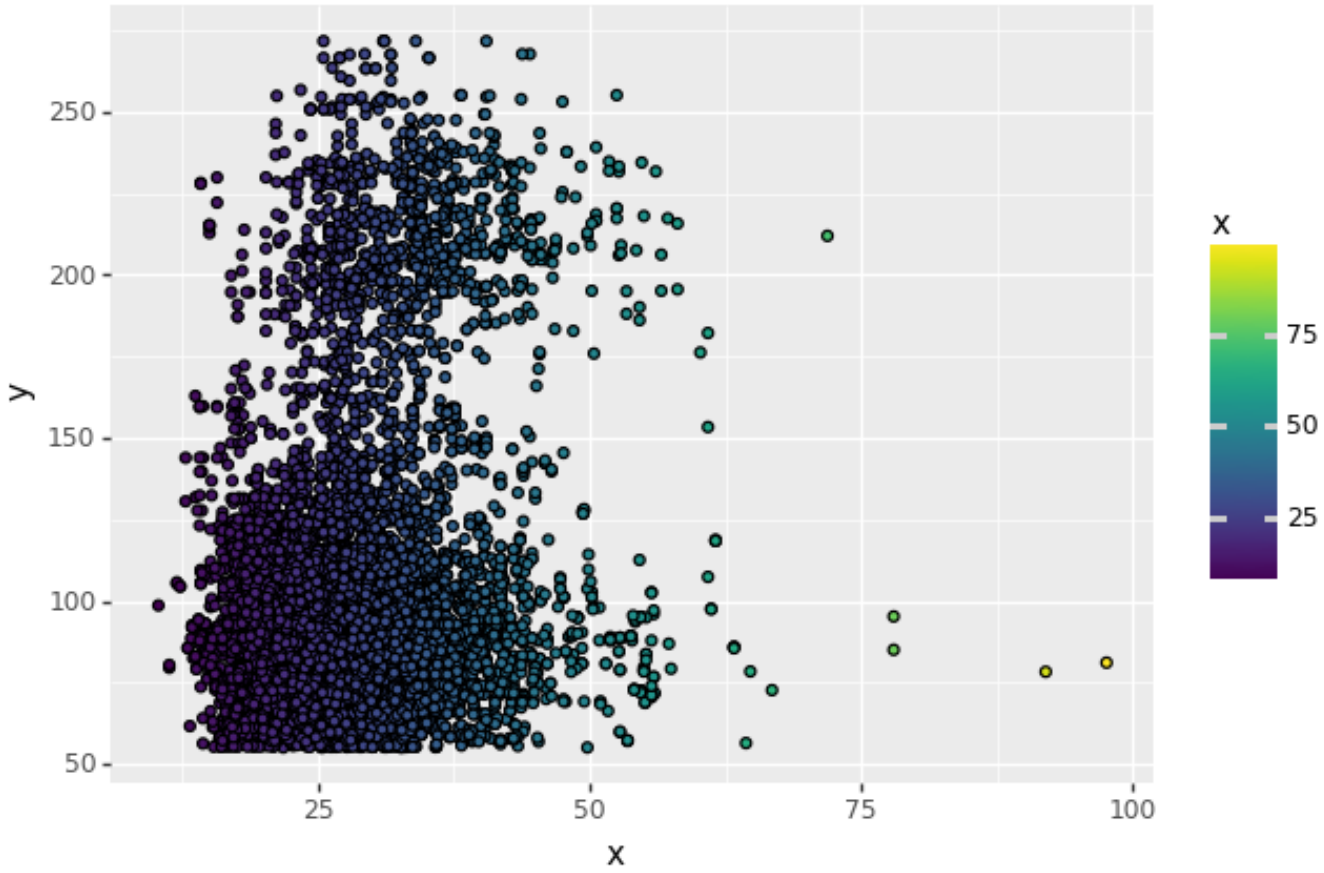



```
<ggplot: (8775091691165)>
```

The scatter plot shows the majority of pe

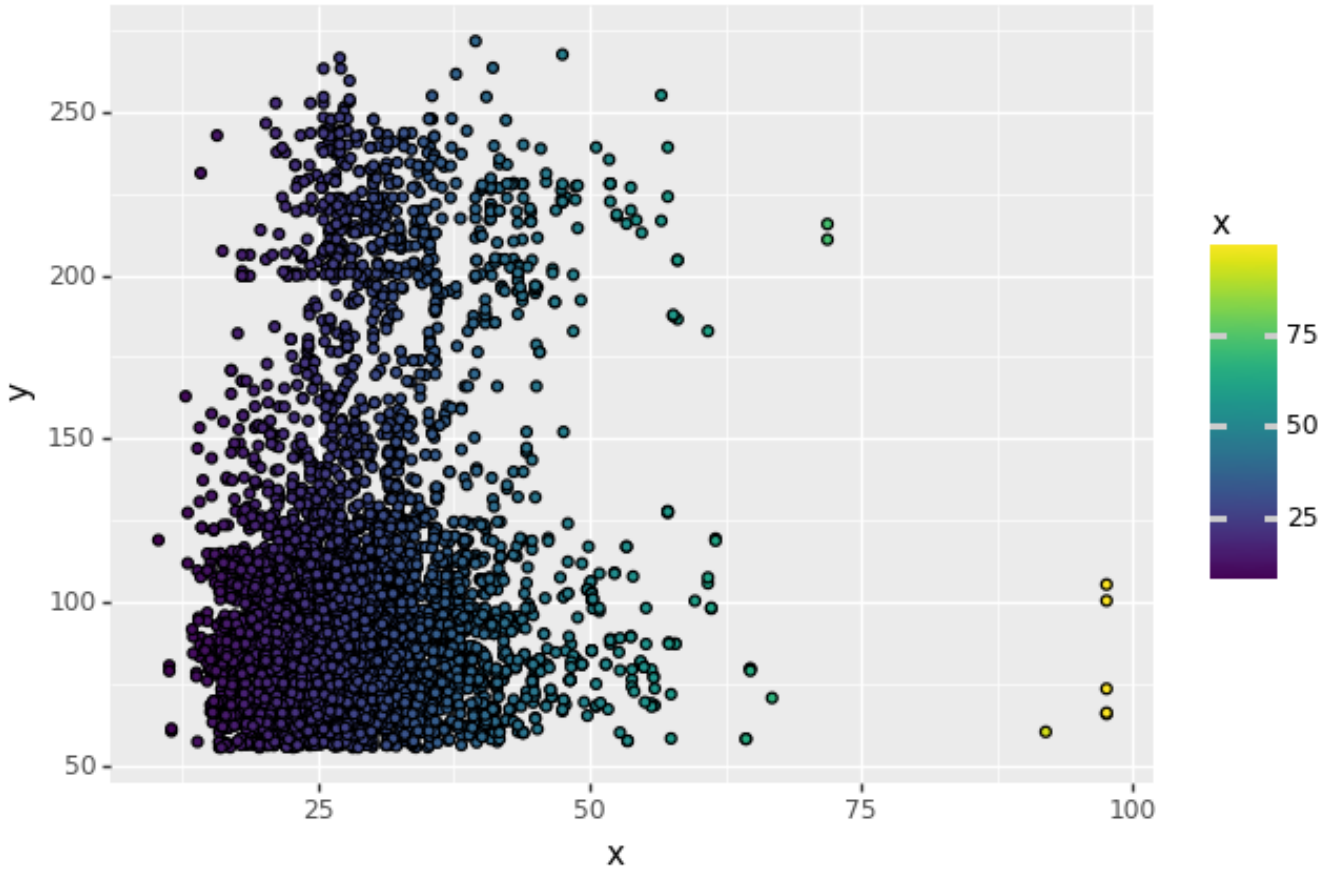
The scatter plot shows the majority of people

```
vote = DF[DF["reg_to_vote"] == 1.0]
DF4 = pd.DataFrame({"x": vote["bmi"], "y": vote["avg_glucose_level"]})
DF4
ggplot(DF4, aes(x="x", y="y", fill = "x")) + geom_point()
```




```
<ggplot: (8775091535409)>
```

```
no_vote = DF[DF["reg_to_vote"] == 0.0]
DF5 = pd.DataFrame({"x": no_vote["bmi"], "y": no_vote["avg_glucose_level"]})
DF5
ggplot(DF5, aes(x="x", y="y", fill = "x")) + geom_point()
```




```
<ggplot: (8775091802301)>
```

3. Is your logistic regression model most accurate for people who make less than 30k, between 30-90k, or over 90k? Discuss the potential accuracy and ethical implications if your model were more accurate for different groups (you can use the full model from part I-3 to check accuracy for each of these groups; DO NOT create/fit new models for each income range, use the model from part I-3 to calculate the accuracy for each of these groups.)

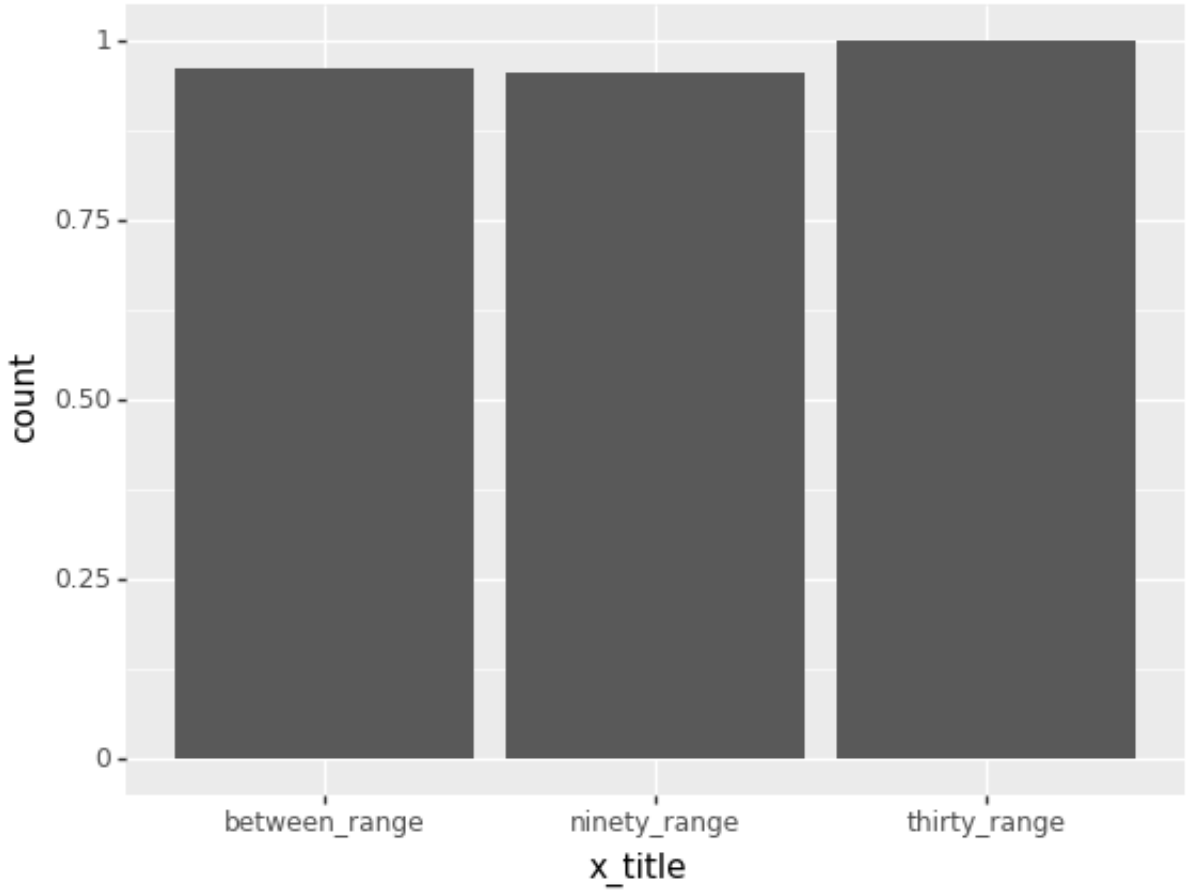
```
thirty_range = DF[DF["income_in_k"] < 30]
between_range = DF[DF["income_in_k"] > 30]
between_range = between_range[between_range["income_in_k"] < 90]
ninety_range = DF[DF["income_in_k"] > 90]

continuous_predictors = ["age", "avg_glucose_level", "bmi","raccoons_to_fight", "fa


thirty_acc =model.score(thirty_range[continuous_predictors], thirty_range['stroke']
between_acc = model.score(between_range[continuous_predictors], between_range['stro
ninety_acc =model.score(ninety_range[continuous_predictors], ninety_range['stroke']
```

```
x_title = ["thirty_range", "between_range", "ninety_range"]
y = [thirty_acc, between_acc, ninety_acc]

DF5 = pd.DataFrame({"y": y, "x_title": x_title})
DF5
ggplot(aes(x="x_title", weight="y"), DF5) + geom_bar()
```




```
<ggplot: (8775091589933)>
```

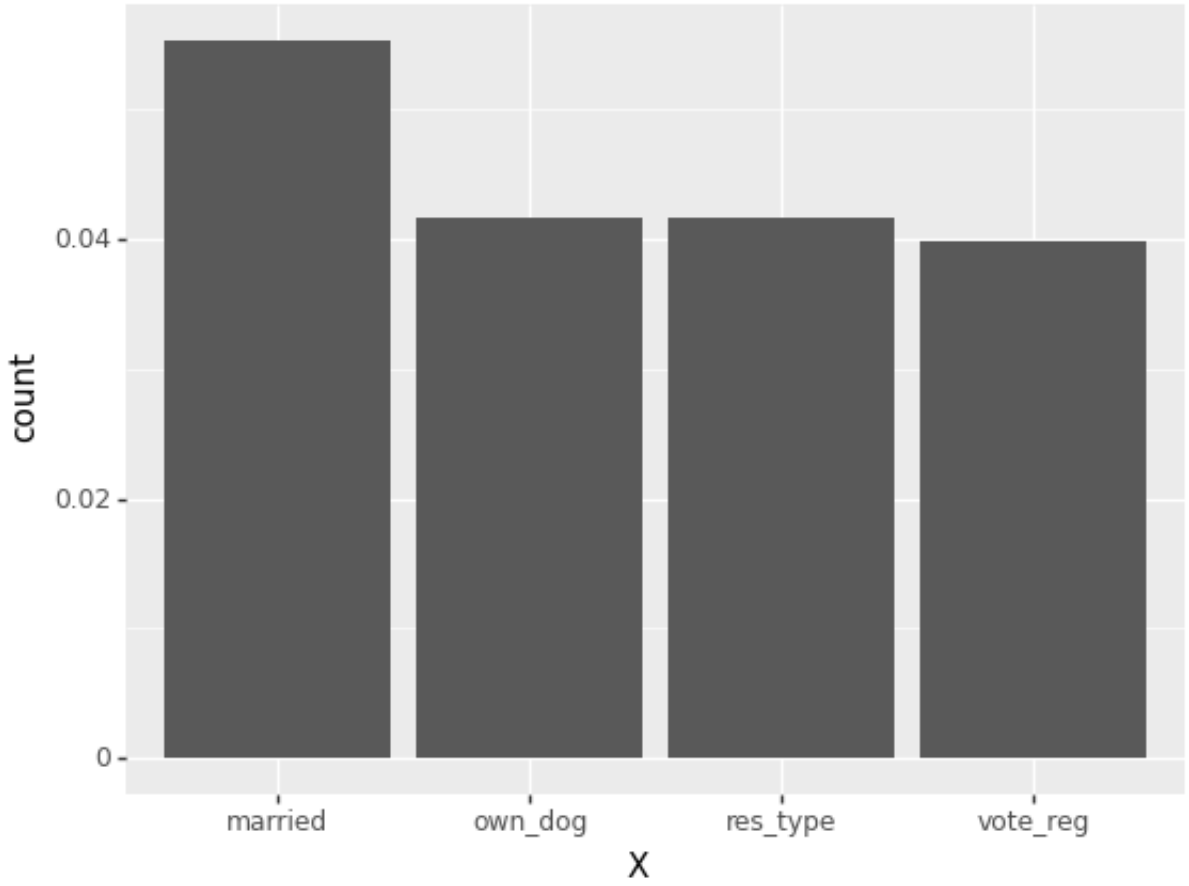
4. Which of the following variables is the strongest predictor of having a stroke (owning a dog, residence type, marriage, being registered to vote)? How were you able to tell?

```
own_dog = DF[DF["dog_owner"] == 1.0]
res_type = DF[DF["Residence_type"] == 1.0]
married = DF[DF["ever_married"] == 1.0]
vote_reg = DF[DF["reg_to_vote"] == 1.0]

own_dog_prob = np.sum(own_dog["stroke"]) / len(own_dog["stroke"])
res_type_prob = np.sum(res_type["stroke"]) / len(res_type["stroke"])
married_prob = np.sum(married["stroke"]) / len(married["stroke"])
vote_reg_prob = np.sum(vote_reg["stroke"]) / len(vote_reg["stroke"])


X = ["own_dog", "res_type", "married", "vote_reg"]
y = [own_dog_prob, res_type_prob, married_prob, vote_reg_prob]
DF6 = pd.DataFrame({"y": y, "X": X})

ggplot(aes(x="X", weight="y"), DF6) + geom_bar()
```




```
<ggplot: (8775091534641)>
```

5. Create a variable er_visits_per_year that calculates the # of visits to the ER that a person has had per year of life. Store this variable in your data frame (no need to include this variable in the previous logistic regression model). Is the # of ER visits per year different for stroke and non-stroke patients? How can you tell?

```
DF["er_visits_per_year"] = DF["er_visits"] / DF["age"]
DF
```

	age	hypertension	heart_disease	ever_married	Residence_type	avg_glucose_
0	60.0	1.0	0.0	0.0	1.0	
1	4.0	0.0	0.0	0.0	0.0	
2	77.0	0.0	0.0	1.0	1.0	
3	37.0	0.0	0.0	1.0	1.0	
4	44.0	0.0	0.0	0.0	0.0	
..	...	...	...	...	...	
995	52.0	1.0	0.0	0.0	1.0	
996	56.0	0.0	0.0	1.0	1.0	
997	60.0	1.0	0.0	1.0	0.0	
998	77.0	0.0	0.0	1.0	0.0	
999	44.0	0.0	0.0	1.0	0.0	

22 rows × 17 columns

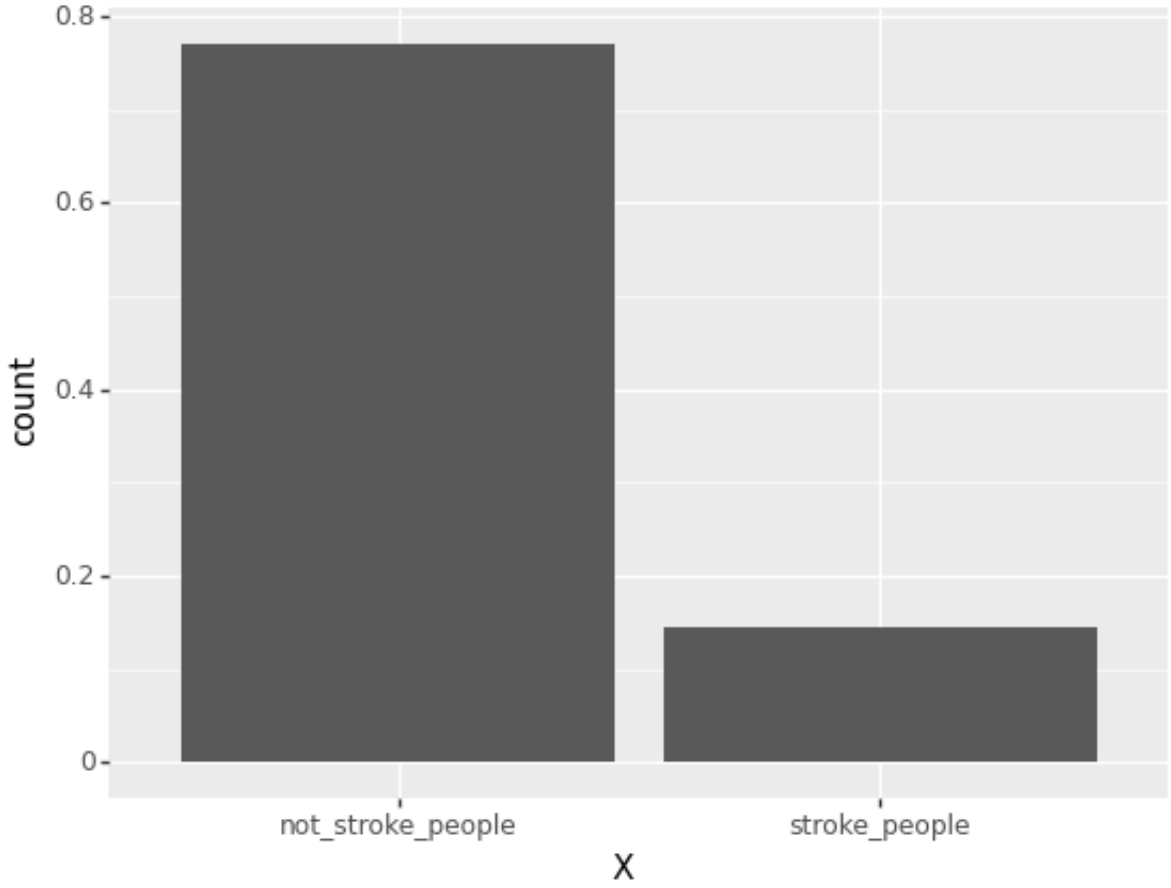
```
stroke_people = DF[DF["stroke"] == 1.0]
not_stroke_people = DF[DF["stroke"] == 0.0]


stroke_people_prob = np.sum(stroke_people["er_visits_per_year"]) / len(stroke_peopl
not_stroke_people_prob = np.sum(not_stroke_people["er_visits_per_year"]) / len(not_
print(not_stroke_people_prob)
```

```
0.7700354208476468
```

```
X = ["stroke_people", "not_stroke_people"]
y = [stroke_people_prob, not_stroke_people_prob]

DF7 = pd.DataFrame({"y": y, "X": X})
ggplot(aes(x = "X", weight = "y"), DF7) + geom_bar()
```




```
<ggplot: (8775091093297)>
```

✓ 0s completed at 2:09 PM