HW1_SP21

March 5, 2021

1 HW1

GENERAL INSTRUCTIONS:

- For all ggplots, make sure you make changes so that the data viz is effective, clear, and does not contain distracting elements.
- CLEARLY mark where you are answering each question.
- Show all code necessary for the analysis, but remove superfluous code

---

1.1 1

Using the dataset linked here, build a linear regression model to predict *reaction time* based on all the other variables.

a) use an 80/20 train test split for model validation and make sure you z score your continuous variables

b) check the linearity assumption for your continuous variables using ggplot. Discuss in detail what you are checking for and what you see for this model.

c) check heteroskedasticity by plotting predicted reaction times/residuals using ggplot. Discuss in detail what you are checking for and what you see for this model.

d) discuss in detail which metrics you use to check your model performance and why, and whether you think your model did well.

Feel free to add cells to this notebook in order to execute the code, but for parts b,c, and d, make sure you put the discussion part in a *Markdown* cell, do not use code comments to answer.

[5]:
```
# import necessary packages
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
from plotnine import *
import statsmodels.api as sm

import sklearn
from sklearn.linear_model import LinearRegression # Linear Regression Model
from sklearn.preprocessing import StandardScaler #Z-score variables
```

```
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score #model␣
 ↪evaluation

from sklearn.model_selection import train_test_split # simple TT split cv
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

%matplotlib inline
```

[6]:
```
# code
DF = pd.read_csv("https://raw.githubusercontent.com/cmparlettpelleriti/
 ↪CPSC392ParlettPelleriti/master/Data/reactionTime.csv")
```

[7]:
```
# checking is DF has missing values
# DF.isnull().sum(axis=0) # all of the entries are filled - no missing data
```

[8]:
```
# predictors (independent variables)
predictors = ["age", "boredom_rating", "risk_propensity", "height",␣
 ↪"left_handed"]
cont_predictors = ["age", "boredom_rating", "risk_propensity", "height"]
```

2 1A)

2.1 Question: Use an 80/20 train test split for model validation and make sure you z score your continuous variables split data: 80/20 split = 80% used to train data, 20% used to test data

[9]:
```
# ****************************************   1A)       ␣
 ↪*****************************************************
X_train, X_test, y_train, y_test = train_test_split(DF[predictors],␣
 ↪DF["reaction_time"], test_size=0.2)

z = sklearn.preprocessing.StandardScaler() # sklearn API for converting data␣
 ↪into z-score form

# Z-score only the continuous X_train and X_test predictor variables
X_train[cont_predictors] = z.fit_transform(X_train[cont_predictors]) # replace␣
 ↪cont_variables with their z-score vals
X_test[cont_predictors] = z.transform(X_test[cont_predictors]) # only transform␣
 ↪bc do not use test to fit model
# prevent data from leaking into our model otherwise test set is not un-seen

LR = LinearRegression() # sklearn API for creating a LR model
LR.fit(X_train, y_train) # fit the X and y training data to the LR model
```

[9]: `LinearRegression()`

3 1B)

3.1 Question: Check the linearity assumption for your continuous variables using ggplot and discuss in detail what you are checking for and what you see for this model.

3.1.1 What is the Linearity Assumption:

- The linearity assumption is the assumption that the relationship between predictors and the outcome is linear.

3.1.2 How will I check for violations of the Linearity Assumption:

- To test for this assumption, I made a scatter plot for each predictor (age, boredom_rating, etc) against the outcome variable, reaction_time. The graphs provide us with insights on how each predictor affects reaction time.

4 1B cont)

4.1 Scatter Plot of Age vs Reaction Time

4.1.1 Graph Description

- The graph below illustrates the relationship between age and reaction time.

4.1.2 What I am checking for:

- I am looking to see if there is a linear pattern. In other words, I am looking to see if on average as age increases/decreases does reaction time increase or decrease linearly.

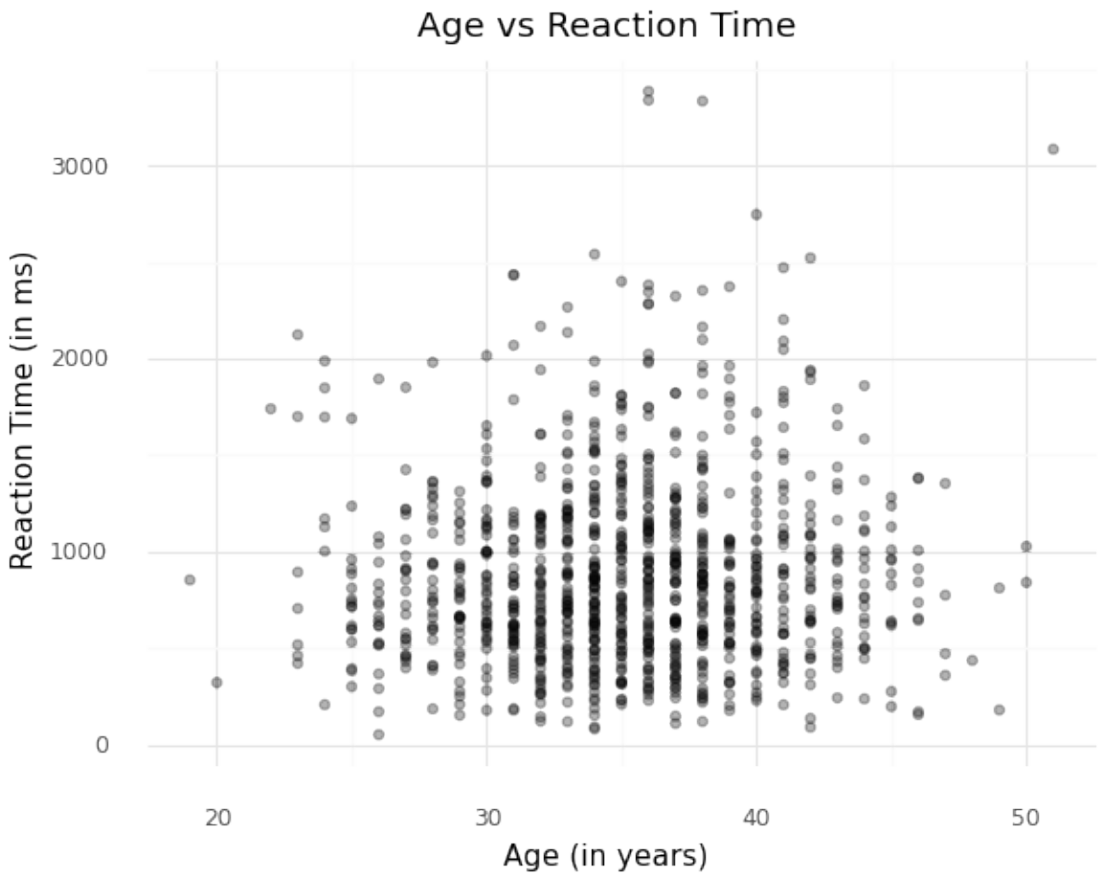
4.1.3 Graph Commentary:

- The spread of the reaction times suggests that age possibly has a positive weak linear relationship to reaction time. In general as age increases, the maximum or highest reaction time is found with the exception of a couple outliers. For example, the highest reaction time at at age 20 it is about 300ms, at age 30 it is about 2000ms, and at age 40 it is about 2700ms. This trend may possibly suggest a positive linear relationship between age and reaction time, however there are outliers. For example, the highest reaction time recorded is nearly 3900ms of an age of about 37 years old. But the highest age is about 52 with a reaction time of about 3200ms.
- In general, there is not too much spread of reaction times across the different ages which helps support the notion that age does affect reaction time. Although I can make these observations, I cannot make any concrete conclusions about the relationship between age and reaction time.

4.1.4 Does age violate the linearity assumption:

- The linearity assumption is not violated with age.

[10]:
```
# Age vs Reaction Time
(ggplot(DF, aes(x = "age", y = "reaction_time")) + geom_point(color = 'black',␣
↪alpha = 0.3) + theme_minimal() + ggtitle("Age vs Reaction Time") + labs(x =␣
↪"Age (in years)", y = "Reaction Time (in ms)"))
```



[10]: <ggplot: (322943763)>

5 1B cont)

5.1 Scatter Plot of Boredom Rating vs Reaction Time

5.1.1 Graph Description:

- The graph below illustrates the relationship between boredom_rating and reaction_time.

5.1.2 What I am checking for:

- I am looking to see if there is a linear pattern/relationship from the graph.

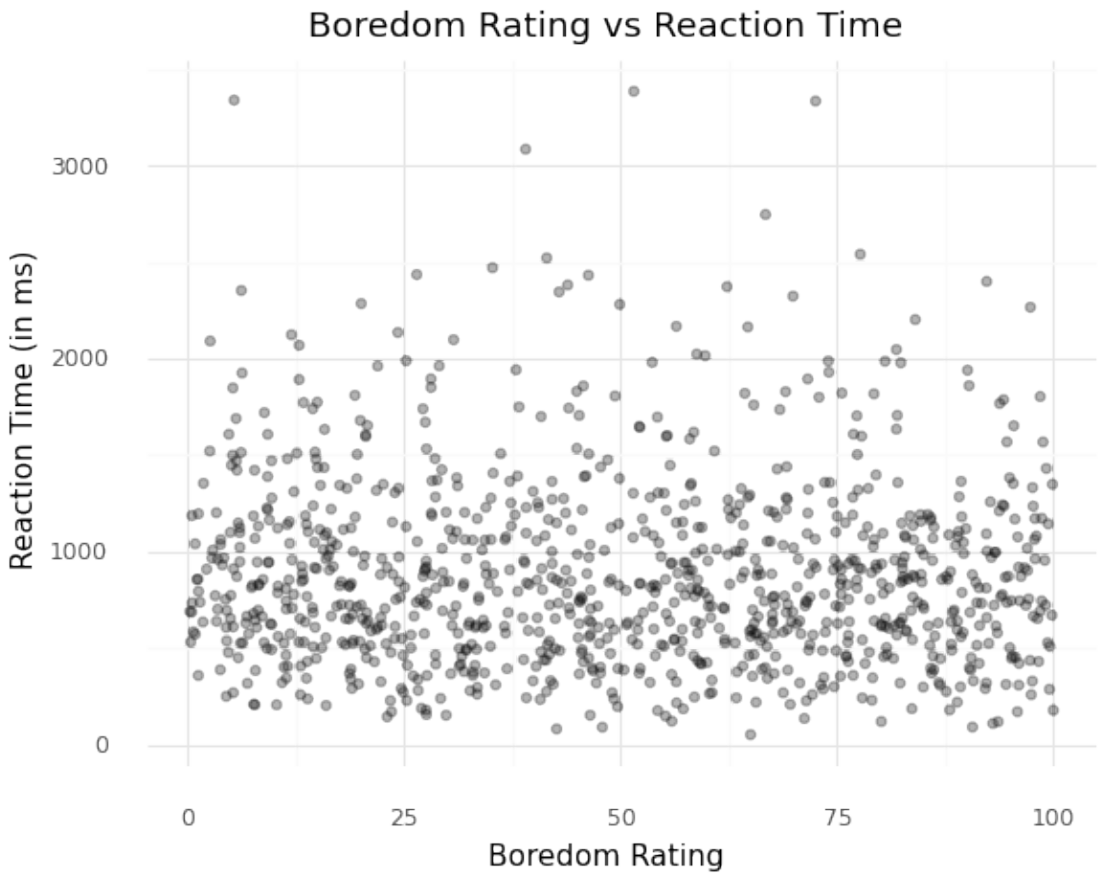
5.1.3 Graph Commentary:

- The distribution of the data points possibly suggests there is no linear relationship between boredom rating and reaction time. The vast majority of reaction times are between 200ms - 1500ms and these majority reaction times are coming from boredom ratings throughout the boredom rating range. Having said that, the increase or decrease of boredom rating does not seem to effect reaction time.
- In general, there is a great amount of spread of reaction times across the different boredome rates which supports the notion that boredom rating does not affect reaction time. Although I can make these observations, I cannot make any concrete conclusions about the relationship between boredom rating and reaction time.

5.1.4 Does boredom rating violate the linearity assumption:

- The linearity assumption is violated with boredom rating.

[11]:
```
# Boredom Rating vs Reaction Time
(ggplot(DF, aes(x = "boredom_rating", y = "reaction_time")) + geom_point(color␣
↪= 'black', alpha = 0.3) + theme_minimal() + ggtitle("Boredom Rating vs␣
↪Reaction Time") + labs(x = "Boredom Rating", y = "Reaction Time (in ms)"))
```



[11]: <ggplot: (325230589)>

6 1B cont)

6.1 Scatter Plot of Risk Propensity vs Reaction Time

6.1.1 Graph Description

- In the graph below, the relationship between risk propensity and reaction time is plotted.

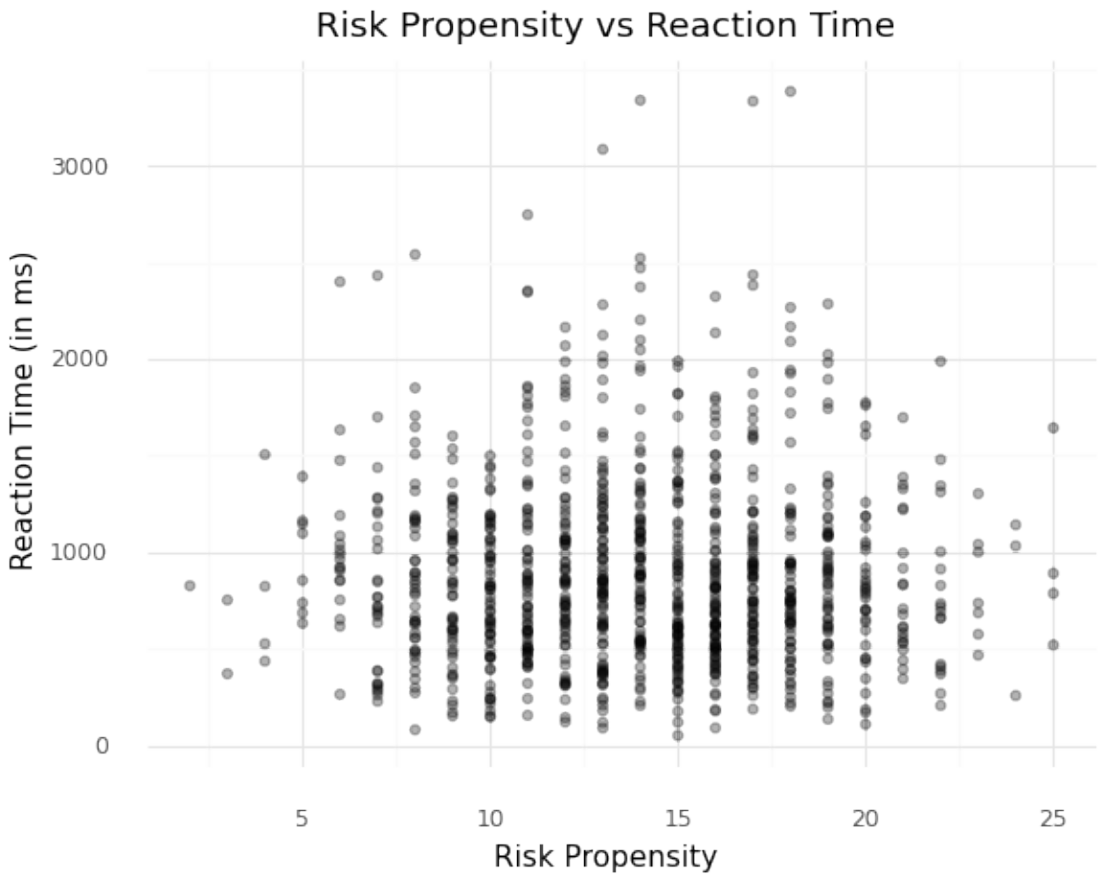
6.1.2 What I am checking for:

- I am looking to see if there is a linear pattern/relationship from the graph. The spread of the reaction times possibly suggests that risk propensity has a positive linear relationship with reaction time but only up to a certain risk propensity. Once that threshold is reached, there seems to be a negative linear relationship. Up to about risk propensity of 17, as risk propensity increases, the maximum or highest reaction time increases with the exception of a couple outliers. For example, the highest reaction time at risk propensity 5 is about 1400ms, at risk propensity 10 it is at about 1500ms, at risk propensity 15 it is about 2000ms and risk propensity 17 it is about 3900ms. The risk propensities after 17, generally have faster and faster reaction times. For example, at risk propensity 19 it has a highest reaction time of 2400ms, at risk propensity 20 the highest reaciton time is about 1800ms, and at risk propensity 25 the highest reaction time is about 1600ms. This pattern suggests a normal distribution relationship between risk propensity and reaction time.
- In general, there is not too much spread of reaction times across the different risk propensities which supports the notion that risk propensity does affect reaction time. Although I can make these observations, I cannot make any concrete conclusions about the relationship between risk propensity and reaction time.

6.1.3 Does risk propensity violate the linearity assumption:

- The linearity assumption is not violated with risk propensity.

[12]:
```
# Risk Propensity vs Reaction Time
(ggplot(DF, aes(x = "risk_propensity", y = "reaction_time")) + geom_point(color␣
↪= 'black', alpha = 0.3) + theme_minimal() + ggtitle("Risk Propensity vs␣
↪Reaction Time") + labs(x = "Risk Propensity", y = "Reaction Time (in ms)"))
```



[12]: <ggplot: (325234966)>

7 1B cont)

7.1 Scatter Plot of Height vs Reaction Time

7.1.1 Graph Description

- In the graph below, the relationship between height and reaction time is plotted.

7.1.2 What I am checking for:

- I am looking to see if there is a linear pattern/relationship from the graph.

7.1.3 Graph Commentary:

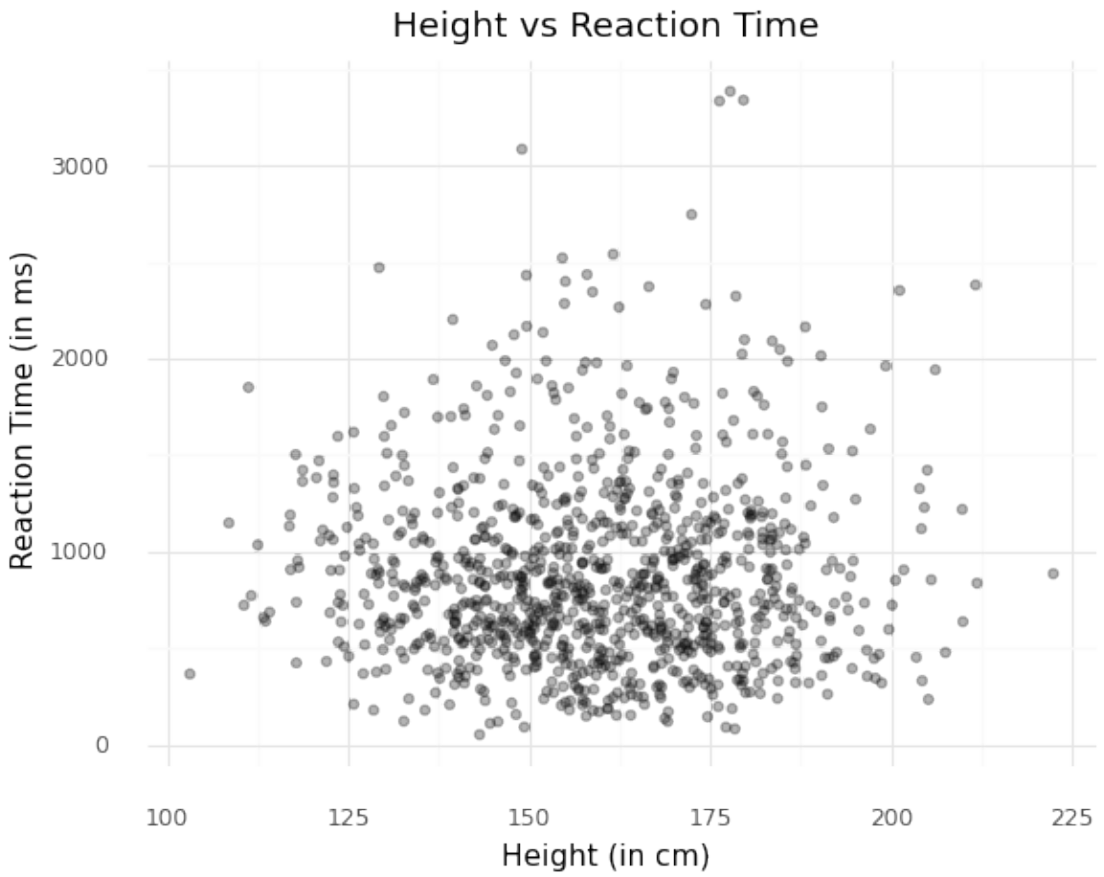
- The spread of the reaction time data points possibly suggest that height does not have a linear relationship with reaction time. The vast majority of reaction times are between 200ms - 1500ms and these majority reaction times are coming from heights between 130 - 180cm. Having said that, the increase or decrease of height does not seem to effect reaction time.

- In general, there is a great amount of spread of reaction times across the different heights which supports the notion that height does not affect reaction time.

7.1.4 Does height violate the linearity assumption:

- The linearity assumption is violated with height.

[13]:
```
# Height vs Reaction Time
(ggplot(DF, aes(x = "height", y = "reaction_time")) + geom_point(color =␣
↪'black', alpha = 0.3) + theme_minimal() + ggtitle("Height vs Reaction Time")␣
↪+ labs(x = "Height (in cm)", y = "Reaction Time (in ms)"))
```



[13]:
```
<ggplot: (325230490)>
```

[14]:
```
# ****************************************   1B)          ␣
↪*****************************************************
reaction_time = y_test # only use the y_test values (NOT all of the y values)␣
↪for actual values
reaction_time_pred = LR.predict(X_test) # predicted outcome values based on the␣
↪X_test values
```

```
# Creating a DF consisting of the predicted reaction times vs the actual␣
↪reaction times
true_vs_pred = pd.DataFrame({"predict": reaction_time_pred, "values":␣
↪reaction_time})
```

8 1B cont)

8.1 Scatter Plot of predicted reaction time values vs real reaction time values

8.1.1 Model Description

- The ggplot graph below is graphing my model's predicted reaction times against the real/true value reaction times. The model predicted the reaction times by using predictor Standard Deviation values based on the 80/20 train/test methodology. This graph gives us an idea of how well our model is doing.

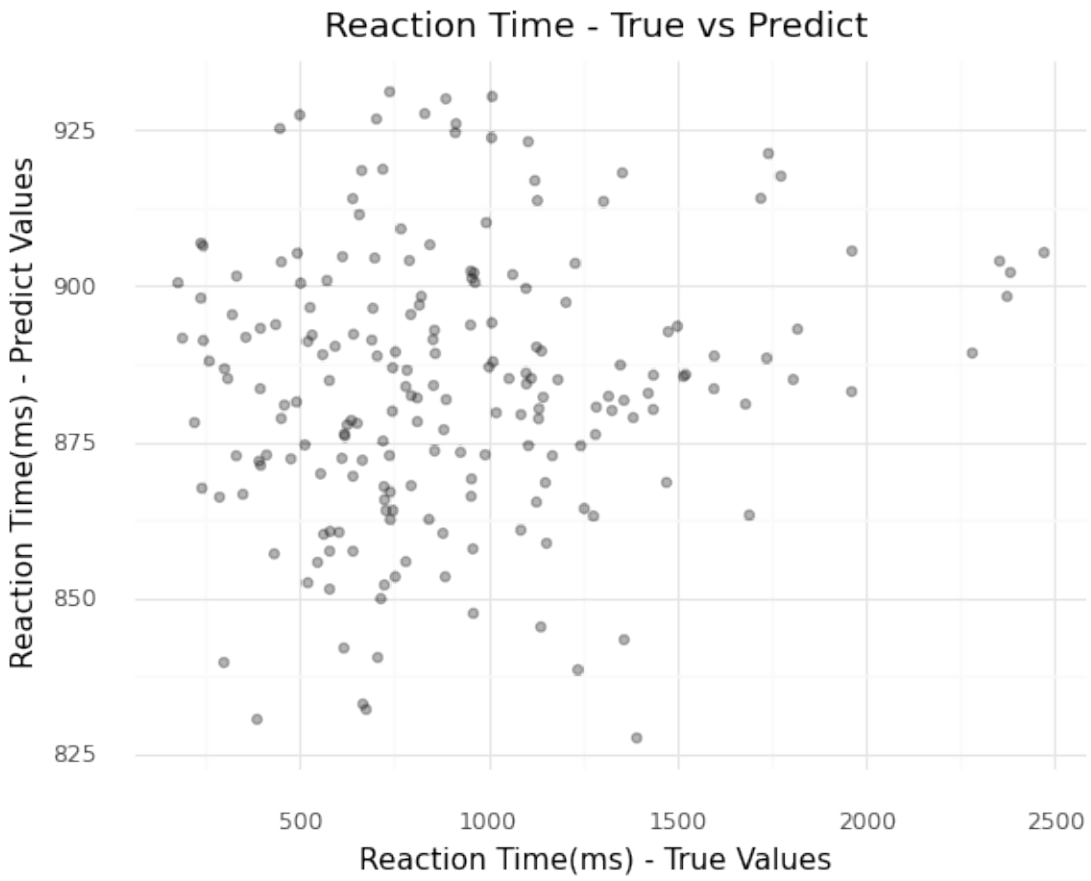
8.1.2 What I am Checking For:

- I am looking to analyze the shape/distribution of the graph/data points to determine the linearity relationship. With an ideal LR model, the relationship between the predictors and outcome is linear. If there is a straight diagnol line, that would indicate that the true values and predicted values are exactly (or almost exactly) the same which would mean our model is doing a great job of predicting reaction times. If the data points are more spread out, that is also an indicator that the model is doing a poor job of predicting the outcome.

8.1.3 Model Interpretation

- The graph below does not have a straight diagnol line and its data points are fairly spread out. One may possibly argue that there is negative sloped diagnol trend from values: [0, 1800] because generally as the values increase in that range, the predicted values decrease as well which may possibly indicate a linear relationship. Although that general trend can be noted, it is not a strong argument because the points in that range are very spread out from each other and thus the Linear Regression Model seems to be doing a poor job of predicting reaction times.

[15]:
```
# ****************************************   1B cont)        ␣
↪*****************************************************
(ggplot(true_vs_pred, aes(x = "values", y = "predict")) + geom_point(color =␣
↪'black', alpha = 0.3) + theme_minimal() + ggtitle("Reaction Time - True vs␣
↪Predict") + labs(x = "Reaction Time(ms) - True Values", y = "Reaction␣
↪Time(ms) - Predict Values"))
# TODO: Add title, make graph look neater, add transparency to help with␣
↪overlapping data
```



[15]: <ggplot: (325279428)>

9 1C)

9.1 Question: Check heteroskedasticity by plotting predicted reaction times/residuals using ggplot. Discuss in detail what you are checking for and what you see for this model.

[16]:
```
# ****************************************   1C)
# ******************************************************
# check heteroskedasticity by plotting predicted reaction times/residuals using ggplot.
# Discuss in detail what you are checking for and what you see for this model.

# residual = true value - predicted value (think mean squared error)

# assump is a DataFrame:
```

```
# x = predicted values, y (columnms) = error (difference between true value and␣
 ↪predicted value)
assump = pd.DataFrame({"error": reaction_time - reaction_time_pred, "predicted":
 ↪ reaction_time_pred})
```

10 1C cont)

10.1 Scatter Plot of predicted values vs errors

10.1.1 Model Description:

- The ggplot model below is a scatter splot depicting the Linear Regression model's predicted reaction time vs its associated residual errors.

10.1.2 What I am checking for:

- The graph gives us insight on the heteroskedasticity of the Linear Regression Model because it maps out the variation of error of the predictions. I am looking to see if the variation of errors across all possible values are consistent or spread out throughout each x-value.

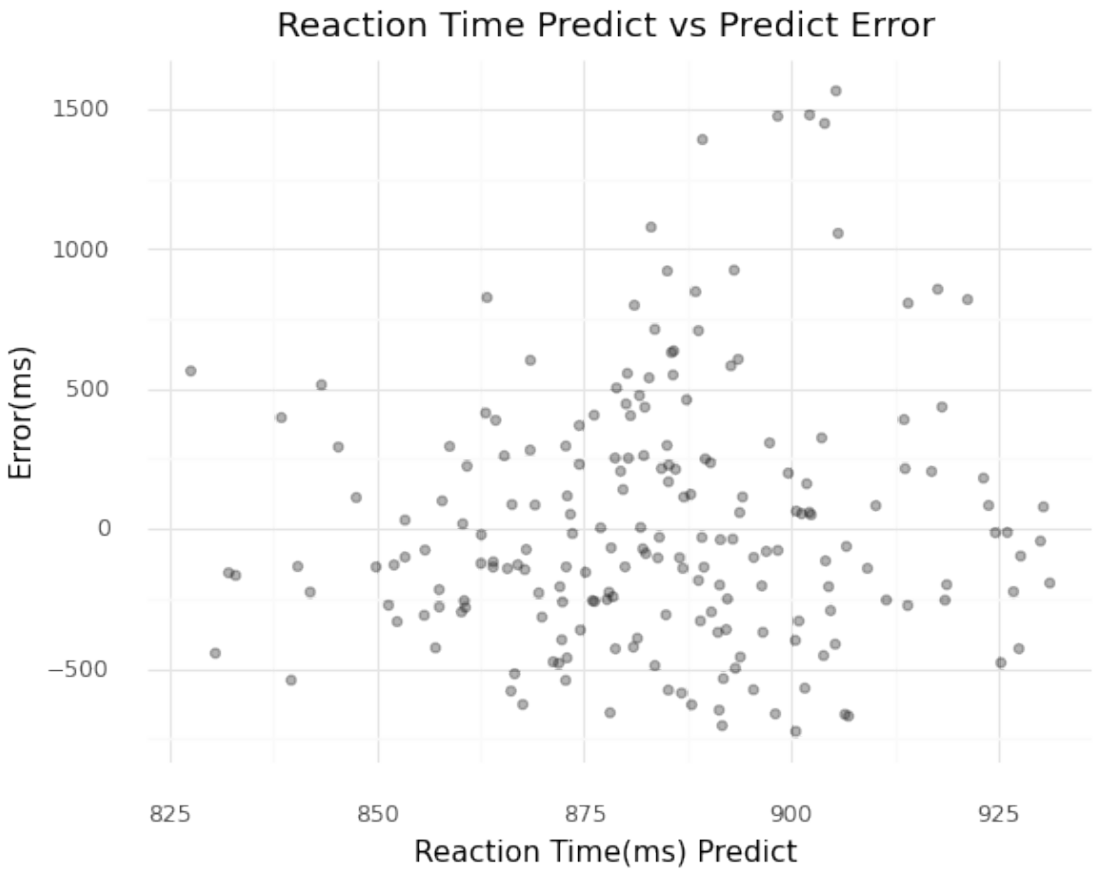
10.1.3 Model Interpretation:

- The graph suggests that the variation in error is somewhat consistent throughout the predictions. For example, at predicted value of 875, there are 5 data points which all have about the same error value. This pattern is seen across some of the other x predicted values, but there many exceptions to this trend. At about x-value of 825, the y error values are very dispersed and in other instances such as the x-value range: [850-900], the y error values are fairly dispersed.

10.1.4 Is heteroskedasticity violated:

- Since the variation in error is only somewhat consistent throughout the predictions, the LR assumption of heteroskedasticity has been violated.

[17]:
```
# ****************************************   1C cont)          ␣
 ↪*****************************************************
# ggplot to check heteroskedasticity
ggplot(assump, aes(x = "predicted", y = "error")) + geom_point(color = 'black',␣
 ↪alpha = 0.3) + theme_minimal() + ggtitle("Reaction Time Predict vs Predict␣
 ↪Error") + labs(x = "Reaction Time(ms) Predict", y = "Error(ms)")
# TODO: Talk more in specific about graph like where in the graph shows the␣
 ↪inconsistencies,etc
```



[17]: <ggplot: (325352927)>

11 1D)

11.1 Question: Discuss in detail which metrics you use to check your model performance and why, and whether you think your model did well.

11.1.1 Metrics can use: sum of squared errors, mean squared error (mse), and r2

11.1.2 Metrics did use: mse and r2

Sum of squared errors background and why I did not use it:

- The sum of squared errors is the summation of each residual line error squared. The reason the residual error values are squared is because squaring covers the whole square area of the error. It is ideal for the sum value to be as small as possible relative the outcome units squared because the smaller it is, the better the LR model is at predicting reaction time.
- Although this metric is valuable, I will not use it to check my Model's performance because it overlaps with mse (discussed below).

Mean squared error background and why I did use it:

- The mean squared error is sum of squared errors divided by the number of data points and is considered a loss function because it is a measure of well a model is doing. The mean squared error value tells us approximately what error value we can expect to get from any data point on the LR model. Like the sum of squared errors, the lower the mean squared error is (relative the outcome units squared), the better the LR model is at predicting reaction time.
- I am using mse as a metric to measure my model's performance because it is a good metric to use to check how close the model's forecasts are to actual results.

r2 background and why I did use it:

- r2 represents the percentage of variance that is explained by the model. The closer the percentage or decimal value of r2 is to 1.0, the more the variation is explained by the model (as opposed to external factors/noises). In constrast, an r2 of 0 or close to 0 is an indicator that the model does a poor job of predicting the outcome because the variance is not explained by the model.
- I am using r2 as a metric to measure my model's performance because it helps me understand the strength of the relationship between the predictor variables and the model in a standard scale (0 - 1).

[18]:
```
# ****************************************   1D cont)
↪******************************************************
# mse calculation
# The average squared error between each data point and the predicted value for␣
↪that data point (average squared distance)
mean_squared_error(reaction_time, reaction_time_pred) # args: [test actual␣
↪values, test values]
```

[18]:
```
203865.01593147276
```

12 1D cont)

12.1 mse interpretation

- Although the mse is high, in 20,000s, that does not necessarily mean my model is performing poorly because the mse is in terms of the outcome units squared. In this LR model, the error is in miliseconds squared because the reaction time is measured in miliseconds. To help get a better idea of how well our model is doing, I calculate r2 next since it gives a more standardized score (between 0 and 1).

[19]:
```
# ****************************************   1D cont)
↪******************************************************
#r2 calculation
r2_score(reaction_time, reaction_time_pred) # different r2 method --> LR.
↪score(X_test, y_test)
```

[19]:
```
0.007963657743381036
```

13 1D cont)

13.1 r2 interpretation

- The r2 absolute value is very small, less than 0.02, which indicates our model is performing very poorly at predicting reaction times because the variation in our model's results are not explained from the model itself.

14 1D cont)

14.1 Conclusion about Linear Regression Model

Having a high mse value and a very low r2 value, I can conclude that the model is performing very poorly at predicting reaction times based on the provided predictors (age, boredom_rating, etc).

14.2 2

Using the data set linked here, build a logistic regression model that predicts whether or not someone fell asleep during the experiment based on all the other variables.

a) use k-fold cross validation and make sure to z score your continuous variables

b) once you have used k-fold to validate your model's performance, fit another separate logistic regression with ALL the data. Grab the coefficients, put them in a data frame, add a column to the data frame with the names of the predictors, and make a bar chart (using ggplot) showing the magnitude of each coefficient. Your coef data frame should look like this:

PredictorName	CoefficientValue
Predictor_1	xx.xx
Predictor_2	xx.xx
…	…
Predictor_N	xx.xx

Where `Predictor_1`, `Predictor_2`…`Predictor_N` are the names of your predictors and `xx.xx`s are your coefficient values.

c) using this graph and the table of coefs generated above, interpret/discuss all your coefficients as if you were explaining the results to a non-data scientist. Include in your explanation, the reason for presenting the coefs in log odds, odds, or probability form.

d) discuss in detail which metrics you use to check your model performance and why, and whether you think your model did well.

Feel free to add cells to this notebook in order to execute the code, but for parts c and d, make sure you put the discussion part in a *Markdown* cell, do not use code comments to answer. CLEARLY mark where you are answering each question.

[20]:
```
# K fold is usually used when the data set is small
# break the data set into K folds, for each of the folds: separate test and␣
 ↪train to calculate the metrics (r2, mse, etc)
```

```
# repeat this K times and then combine results to get the average
```

[21]:
```
# import necessary packages
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import numpy as np
from plotnine import *
import statsmodels.api as sm

from sklearn.linear_model import LogisticRegression # Linear Regression Model
from sklearn.preprocessing import StandardScaler #Z-score variables
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score #model␣
↪evaluation

from sklearn.model_selection import train_test_split # simple TT split cv
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
```

[22]:
```
kDF = pd.read_csv("https://raw.githubusercontent.com/cmparlettpelleriti/
↪CPSC392ParlettPelleriti/master/Data/fellAsleep.csv")
```

15 2A)

15.1 Question: Use k-fold cross validation and make sure to z score your continuous variables

[23]:
```
# code
# ****************************************   2A)       ␣
↪**********************************************************
# use k-fold cross validation and make sure to z score your continuous variables

# separate cont and non cont predictors
predictors = ["age", "boredom_rating", "risk_propensity", "height",␣
↪"left_handed"]
cont_predictors = ["age", "boredom_rating", "risk_propensity", "height"]
X = kDF[predictors] # independent variables
y = kDF["fell_asleep"] # dependent/outcome variable: fell_asleep

# Z-score the cont X data
z = sklearn.preprocessing.StandardScaler()
Xz = z.fit_transform(X[cont_predictors])
```

[24]:
```
# ****************************************   2A cont)
↪*****************************************************
# create model
kf = KFold(n_splits = 10) # k = 10 ==> 10 folds

LR = LogisticRegression() # sklearn API LR model
mse = [] # list containing mse of each K-fold
r2 = [] # list containing r2 of each K-fold

for train, test in kf.split(X):
    X_train = X.iloc[train]
    X_test = X.iloc[test]
    y_train = y[train]
    y_test = y[test]

    #model
    model = LR.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse.append(mean_squared_error(y_test, y_pred))
    r2.append(r2_score(y_test, y_pred))
```

16 2B)

16.1 Question: Fit another separate logistic regression with ALL the data. Grab the coefficients, put them in a data frame, add a column to the data frame with the names of the predictors, and make a bar chart (using ggplot) showing the magnitude of each coefficient

[25]:
```
# ****************************************   2B)
↪*****************************************************
coefficients = pd.DataFrame({"Coef": LR.coef_[0], "Name": predictors})
```

[26]:
```
coefficients["Odds Coefs"] = np.exp(coefficients["Coef"])
```

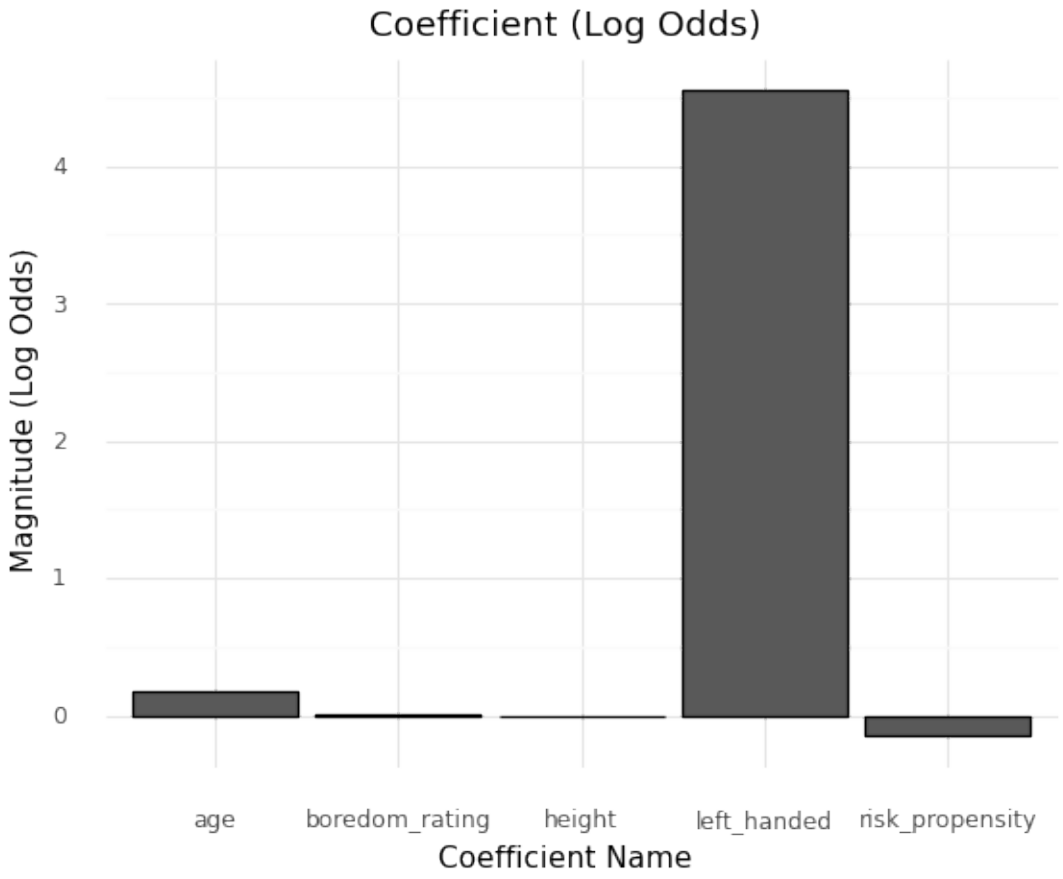
17 2B cont)

17.1 Bar Charts showing the Magnitude of each Coefficient

17.1.1 1st Bar Chart coefficient values are in terms of Log Odds

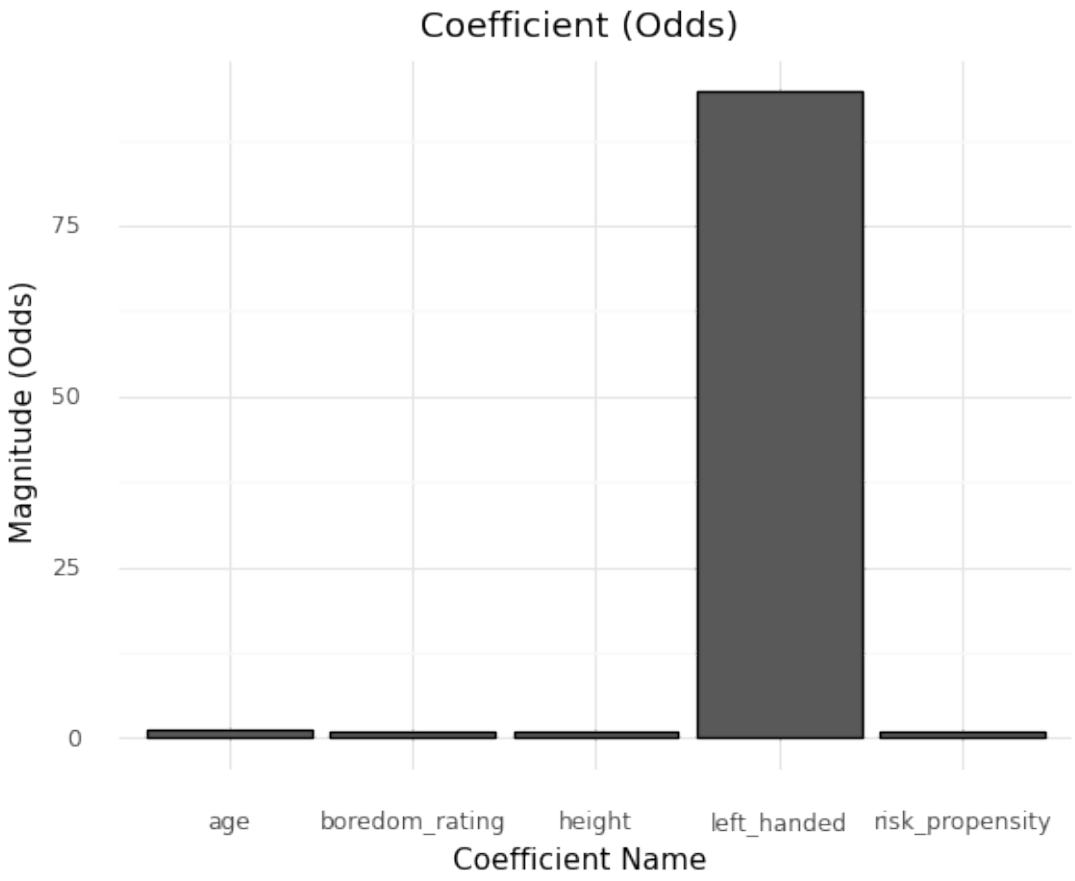
17.1.2 2nd Bar Chart coefficient values are in terms of Odds

[78]:
```
# ****************************************   2B cont)
↪*****************************************************
(ggplot(coefficients, aes(x = "Name", y = "Coef")) + geom_bar(stat =␣
↪"identity", color = "black") + theme_minimal() + ggtitle("Coefficient (Log␣
↪Odds)") + labs(x = "Coefficient Name", y = "Magnitude (Log Odds)"))
```



[78]:
```
<ggplot: (324376540)>
```

[79]:
```
(ggplot(coefficients, aes(x = "Name", y = "Odds Coefs")) + geom_bar(stat =␣
↪"identity", color = "black") + theme_minimal() + ggtitle("Coefficient␣
↪(Odds)") + labs(x = "Coefficient Name", y = "Magnitude (Odds)"))
```



[79]: <ggplot: (324354645)>

18 2C)

18.1 Question: using this graph and the table of coefs generated above, interpret/discuss all your coefficients as if you were explaining the results to a non-data scientist. Include in your explanation, the reason for presenting the coefs in log odds, odds, or probability form.

18.2 Interpretation of Coefficients (Odds)

18.2.1 The dataframe of coefficients created called "coefficients" has a column titled "Coef" which is in terms of Log Odds. To put the coefficients more in perspective for non-data scientists to understand, I converted the Log Odd values to Odds in a new column titled "Odds Coefs". I converted Log Odds to Odds because the general population does not have a good understanding of what log odds means, however, people generally understand what odds means. For example, a person knows that the odds of getting heads or tails from flipping a coin is equal (assuming no exteneral factors are alternating results). I did not convert the coefficient values to probabilities because probability is not constant. In other words, probability does not have a constant relationship with predictor variables and so they should not be used in this context to draw conclusions about the coefficients.

18.2.2 Odd Coefficient Values greater than 1

- In general if the coefficent (in terms of Odds) has a value greater than 1, then that means that the predictor variable has a positive relationship with respect to falling asleep (outcome variable). To be more precise, an increase of a 1 Standard Deviation unit of the predictor variable means that the odds of falling asleep is the coefficient value times more likely to fall asleep.

Age

- An increase in 1 Standard Deviation unit of age means that person is 1.195x more likely to fall asleep.

Boredom Rating

- An increase in 1 Standard Deviation unit of boredom rating means that person is 1.012x more likely to fall asleep. #### Left Handed
- An increase in 1 Standard Deviation unit of being left-handed means that person is 94.637x more likely to fall asleep.

18.2.3 Odd Coefficient Values less than 1

- In general if the coefficent (in terms of Odds) has a value less than 1, then that means that the predictor variable has a negative relationship with respect to falling asleep (outcome variable). To be more precise, an increase of a 1 Standard Deviation unit of the predictor variable means that the odds of falling asleep is the coefficient value times more likely to fall asleep.
- Since the multiplying value is less than 1, the likelihood of falling asleep decreases.

Risk Propensity

- An increase in 1 Standard Deviation unit of risk propensity means that person is 0.861x less likely to fall asleep, which means the person's likelihood of falling asleep actually decreases.

Risk Propensity

- An increase in 1 Standard Deviation unit of height means that person is 0.997x less likely to fall asleep, which means the person's likelihood of falling asleep actually decreases.

19 2D)

19.1 Question: Discuss in detail which metrics you use to check your model performance and why, and whether you think your model did well.

20 2D cont)

20.0.1 Metrics can use: sum of squared errors, mean squared error (mse), and r2

20.0.2 Metrics did use: mse and r2

Sum of squared errors - why I did not use it:

- Although this metric is valuable (for reasons stated in 1D), I will not use it to check my Model's performance because it overlaps with mse (mse discussed below).

Mean squared error - why I did use it:

- I am using mse as a metric to measure my model's performance because it is a good metric to use to check how close the model's forecasts are to the actual results.

r2 - why I did use it:

- I am using r2 as a metric to measure my model's performance because it helps me understand the strength of the relationship between the predictor variables and the model in a standard scale (0 - 1). If the r2 value is low then I know that the variation in my model is not being explained by model, which is an indicator that the model does a poor job of predicting the outcome variable.

[81]:
```
np.mean(mse)
```

[81]:
```
0.097
```

21 2D cont)

21.1 mse interpretation

- In this LR model, the error is in binary units squared because falling asleep is measured in binary units (0 = did not fall asleep, 1 = did fall asleep). Having said that, the mse value is not very low in terms of its units, which most likely means my model is performing moderately well (not great, but also not terrible).

[54]:
```
np.mean(r2)
```

[54]:
```
0.2981561854962839
```

22 1D cont)

22.1 r2 interpretation

- The r2 value, 0.298, is on the lower end of the r2 scale (0 - 1). This r2 value suggests that the model's variation is less 50% explained from the model itself, which is not a good thing for predictive models.

23 2D cont)

23.1 Conclusion about Logistic Regression Model

Having a moderate mse value and a low r2 value, I can conclude that the model is performing poorly at predicting falling asleep based on the provided predictors (age, boredom_rating, etc).