Linear Regression

Mentor: Pararawendy Indarjo

Hey I’m,

Pararawendy Indarjo

I am a,

- CURRENTLY | **Senior DS at Bukalapak**
- 19 - 20 | **Data Analyst at Eureka.ai**

BSc Mathematics

MSc Mathematics

Linkedin :
https://www.linkedin.com/in/pararawendy-indarjo/
Blog : medium.com/@pararawendy19

Outline

- Introduction to Linear Regression
- Linear Regression with Python
    - Single predictor
    - Multiple predictors
- Model diagnostic
- Model evaluation

Types of ML

RECAP

Based on target availability

01

1. Supervised learning

- Supervised = ground truth provided
- I.e. target is available in training data
  - (x1,y1), (x2,y2), (x3,y3), etc..
- Objective: to correctly predict y from unseen data x
- Based on target variable type, can further be divided:
  - Regression
  - Classification

02

2. Unsupervised learning

- Unsupervised = *no* ground truth provided
- I.e. target is NOT available in training data
  - (x1), (x2), (x3), …
- Objective: to find hidden patterns in unlabeled data
  - Clustering
  - Dimensionality reduction

Regression Example

RECAP

Consider a model to predict sales omzet (GMV) using different advertising channels

Features/Predictors

GMV = 0.7 + 2.1 socmed + 3.3 onmedia + 1.4 DOOH

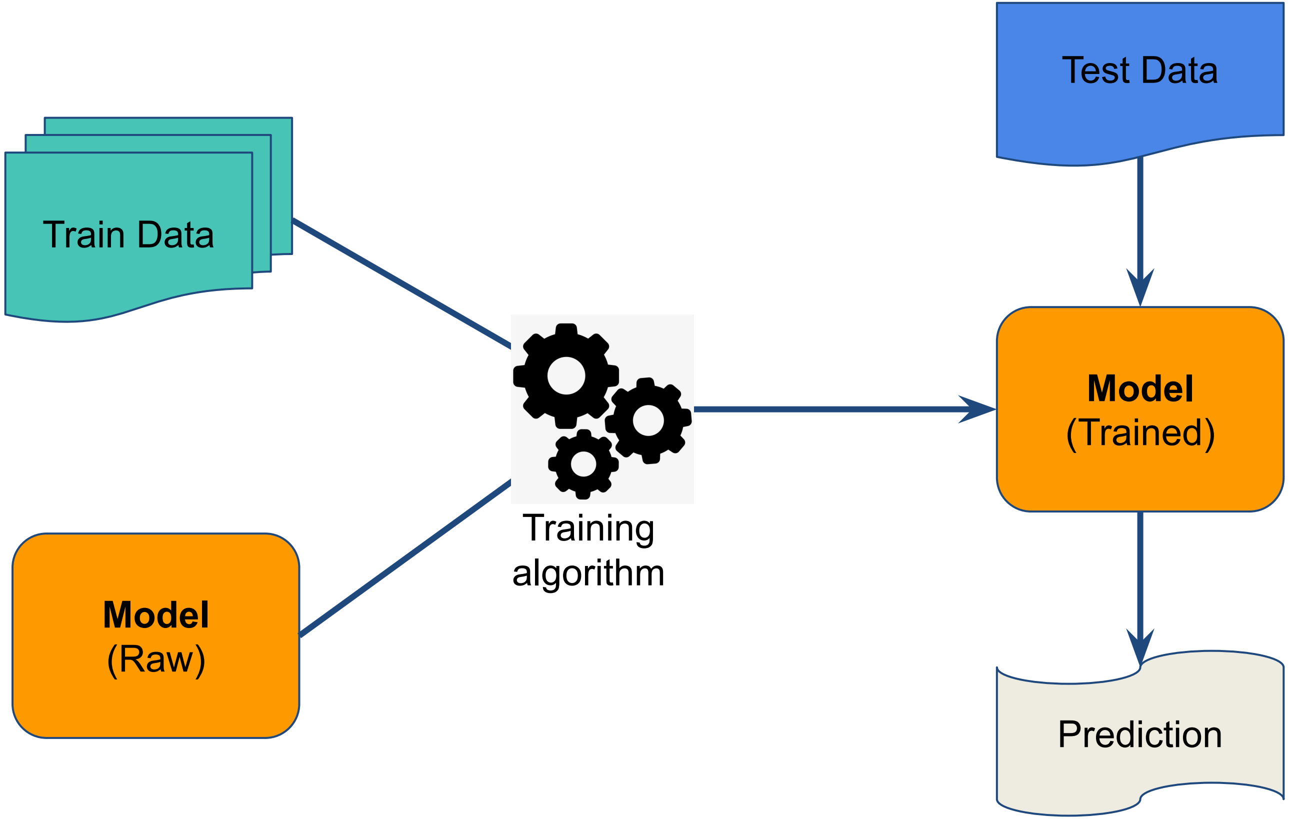
Parameters/Coefficients

Target

- Regression model:
  - Predict GMV (Sales omzet)
  - Predictors are different advertising channels

First: What is a model?

RECAP



Introduction to Linear Regression

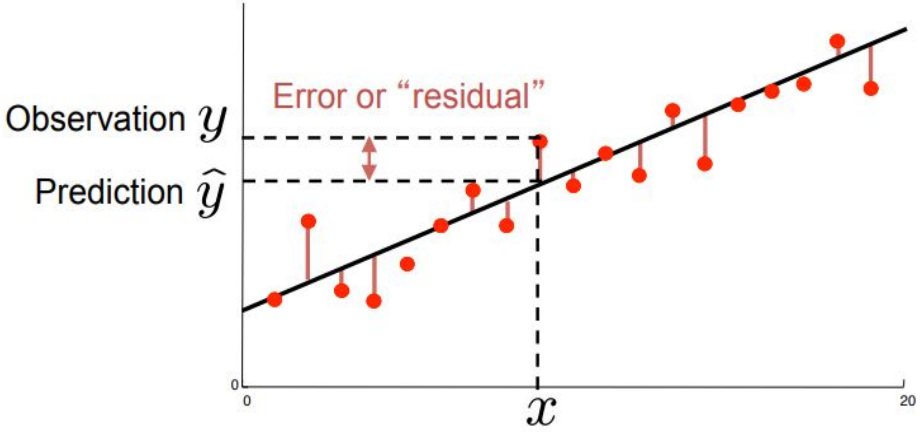
- Linear regression is the oldest statistical model, invented by Legendre in 1805
- On a high-level, linear regression is all about finding a **straight line (linear)** that fits the data most nicely
- The mathematical form

$$y = b_0 + b_1 x_1 + b_2 x_2 + ... + b_n x_n$$

- y is target, x_1, x_2, .. x_n are **predictors/features**, b_0, b_1, b_2, …, b_n are **parameters/coefficients** to learn
- Characteristics
    - Supervised learning (y is provided)
    - Regression model (y is quantitative/numeric)

Residual

- **Residual**: delta between the real value of target variable *y* and the predicted value target variable y_{hat}
    - Predicted values = regression line



- **red points** are our data points
- **black line** is the regression line (our prediction values)

Residuals construct Loss Function

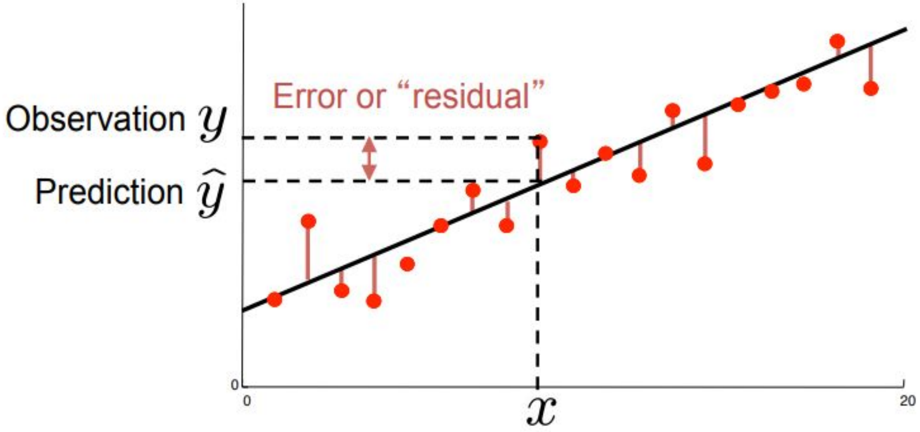
The magnitude we want to minimize when training the model

- **Residual**: delta between the real target variable *y* and the predicted target variable y_{hat}
  - Predicted values = regression line

- Train the model to minimize loss function: **Residual sum of squares**

$$\text{total error} = \sum_i (y_i - \hat{y_i})^2$$

- **Note:** training == finding coefficients

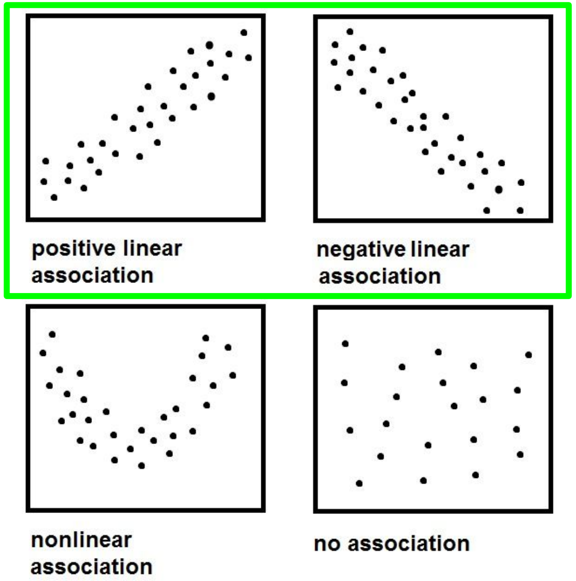


- **red points** are our data points
- **black line** is the regression line (our prediction values)

4 Assumptions of Linear Regression

Assumption 1

- There is a straight-line relationship between the predictors and the target

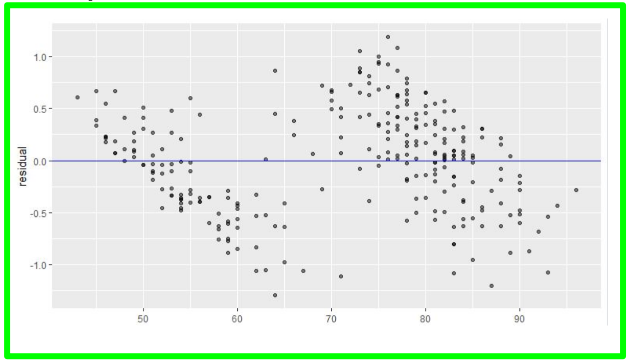


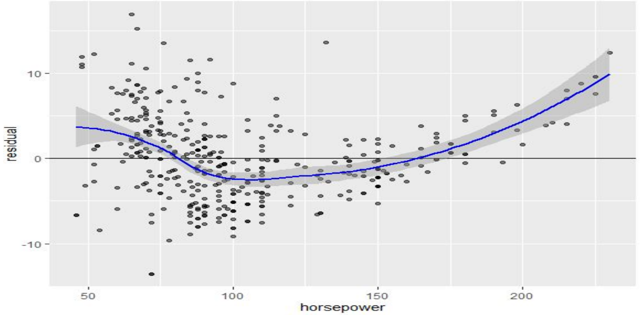
- How do we check this?
  - One predictor: Scatter plot (like above)
  - Multiple predictors: Residual plot

Assumption 1 (cont'd)

Checking linearity assumption via residual plot:

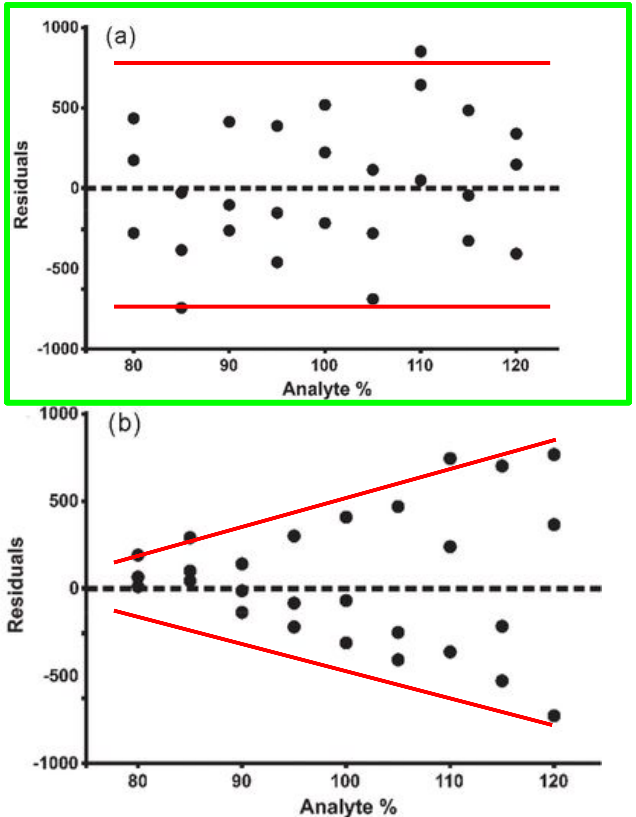
- Validated if there is NO discerning non-linear pattern in residual plot





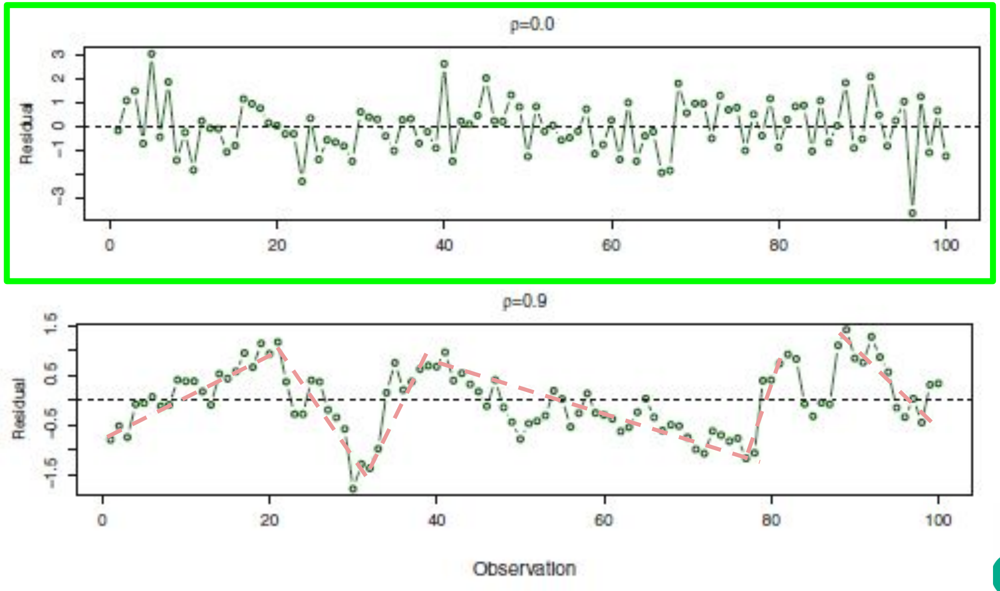
Assumption 2

- Residuals have a constant variance



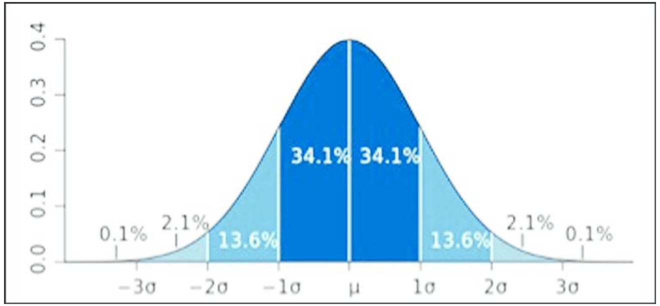
Assumption 3

- Uncorrelated residuals between different observations
- This is equivalent with independent observations

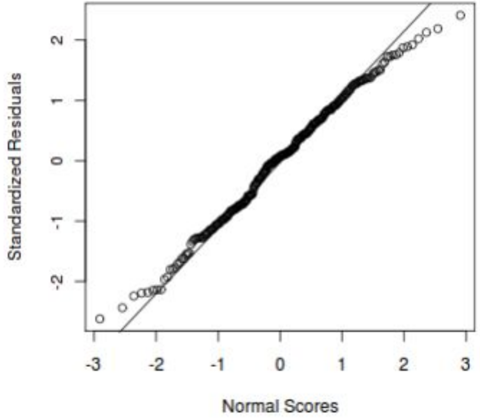


Assumption 4

- Residuals are normally distributed



- Can check using QQ-plot



Remark on Assumptions

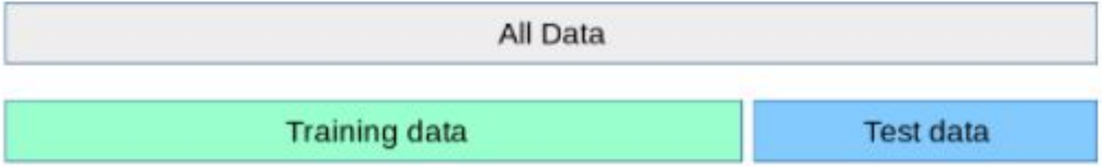
- On real dataset, it is QUITE RARE all of those 4 assumptions are met
- It is the reason why linear regression is usually underperformed on real world data
- Nevertheless, **just carry on!** Real benefits of linear regression:
    - As a baseline model
    - Highly interpretable model

Any

Question

Essential rule in ML modelling

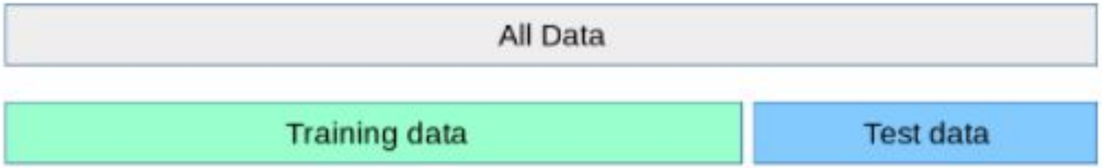
- We are NOT allowed to use ALL of our data to train our model
- Instead, we need to dedicate some portion of our data to become test data



Essential rule in ML modelling

- We are NOT allowed to use ALL of our data to train our model

- Instead, we need to dedicate some portion of our data to become test data



- Because we want our ML model to generalize well on new data
    - I.e. perform well on unseen data

- **Test data is meant to mimic the unseen data**

Linear Regression with Python

- **High level steps:**
  - Split data: training-testing
  - Prepare as numpy arrays (`X_train, y_train;X_test, y_test`)
  - Define and train the model on the training data:
    - Define model: `linreg = LinearRegression()`
    - Train model: `linreg.fit(X_train, y_train)`
  - Interpret and pre-evaluate the model (model diagnostic)
  - Using the trained model to predict test data
    - `linreg.predict(X_test)`
  - Evaluate the model
    - Using various regression metrics

Example: Simple Linear Regression

Regression with only 1 predictor

- We will use `faithful.csv` data
  - `eruptions:` Eruption time (in mins)
  - `waiting:` Waiting time to next eruption (in mins)

- We regress `eruptions` using `waiting`

- `faithful.head()`

	eruptions	waiting
0	3.600	79
1	1.800	54
2	3.333	74
3	2.283	62
4	4.533	85

Example: Simple Linear Regression

Splitting data

```
# split train test
from sklearn.model_selection import train_test_split

feature = faithful.drop(columns='eruptions')
target = faithful[['eruptions']]

ftr_train, ftr_test, tgt_train, tgt_test = train_test_split(feature,
                                                            target,
                                                            test_size=0.20,
                                                            random_state=42)

# set as numpy arrays
X_train = ftr_train.to_numpy()
y_train = tgt_train.to_numpy()
```

We use 80:20 split

For reproducibility

Example: Simple Linear Regression

Model Training and Obtained Coefficients

```
from sklearn.linear_model import LinearRegression

# define the model
simple_reg = LinearRegression()

# train the model
simple_reg.fit(X_train, y_train)
```

	feature	coefficient
0	intercept	-1.939198
1	waiting	0.076638

```
eruptions = −1.939 + 0.076 waiting
```

- -1.939: when waiting is 0, the expected value of eruptions is -1.939
- 0.076: an increase of 1 on waiting time is associated with an increase of 0.076 on eruptions

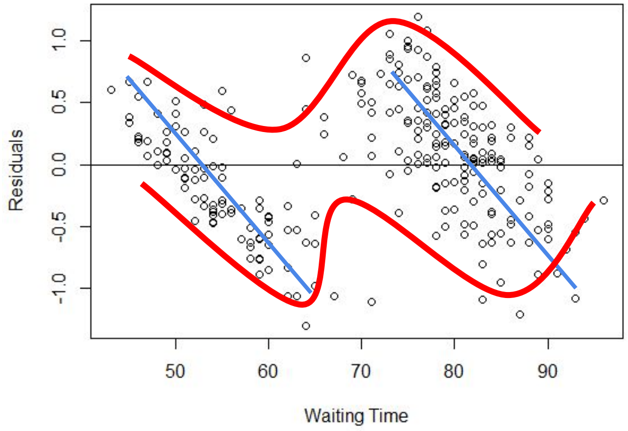
Residual Plot

To check three assumptions of linear regression

- For model with 1 predictor (feature):
  - x-axis = feature value
  - y-axis = residual value

```
# residual plot
sns.scatterplot(data=df_resid, x="x_axis", y="residual")
plt.axhline(0)
plt.show()
```

- Assumptions to check via residual plot:
  - Linear relationship ✔️
  - Constant variance ❌ (look at red curves)
  - Independent observations ❌ (look at blue lines)



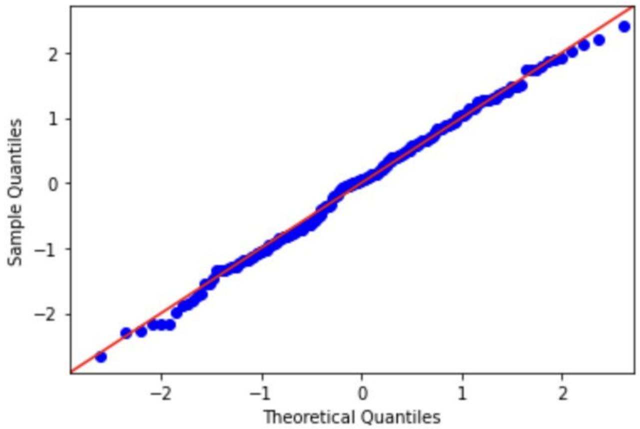
QQ-Plot

To check normality assumption of the residuals

```
# QQplot
from sklearn.preprocessing import StandardScaler

std_resid = StandardScaler().fit_transform(residual.reshape(-1,1))
std_resid = np.array([value for nested_array in std_resid for value in nested_array])

import statsmodels.api as sm
sm.qqplot(std_resid, line='45')
plt.show()
```



- QQ plot for residual
    - To check normality assumption ✔

Any

Question

Hands-On

- Open today's Jupyter notebook on your Google Colab!
- Make sure you have uploaded the required CSV files to your google drive
    - Remember the file path!

The Data Used

- For the rest of the lecture, we will use `regression_data.csv`

- The data is about university admit probability, based on several applicant's features
  - GRE score
  - TOEFL score
  - University ranking
  - Motivation letter quality
  - Recommendation letter strength
  - GPA
  - Research experience

Modeling Flow for >1 predictor

1. **Split the data**
   - 80 : 20 is fine

2. **Multicollinearity check**
   - Calculate VIF score for each feature
   - Correlation analysis to drop redundant feature

3. **Fit the model on training data**
   - Define and fit `LinearRegression()`
   - Only include retained features from step 2

4. **Model diagnostic**
   - Residual plot
   - QQ plot
   - R2 score on training data

5. **Evaluate the model on test data**
   - RMSE

Split the Data

Using train_test_split() function

- Recall, we want to predict `admit_prob`

```
# split train test
from sklearn.model_selection import train_test_split

feature = admit.drop(columns='admit_prob')
target = admit[['admit_prob']]

ftr_train, ftr_test, tgt_train, tgt_test = train_test_split(feature,
                                                            target,
                                                            test_size=0.20,
                                                            random_state=42)
```

Multicollinearity (1/2)

Variance Inflation Factor

- Multicollinearity: when two or more features/predictors are **highly correlated** with each other

- This can cause coefficients of estimates become unreliable
  - i.e. little change in the training data leads to completely different learned coefficients

- We can detect this by computing **Variance Inflation Factor (VIF)** for each feature

$$\mathrm{VIF}(\hat{\beta}_j) = \frac{1}{1 - R^2_{X_j|X_{-j}}}$$

  - On a high level: **VIF feature j will be high if j can be predicted using the rest of other features**. Vice versa
  - VIF == 1 → No multicollinearity
  - VIF between 4 and 10 → Moderate multicollinearity
  - VIF > 10 → Severe multicollinearity

Multicollinearity (2/2)

Using statsmodels library

```
from statsmodels.stats.outliers_influence import variance_inflation_factor as vif
from statsmodels.tools.tools import add_constant

X = add_constant(feature_admit_train)

vif_df = pd.DataFrame([vif(X.values, i)
                for i in range(X.shape[1])],
               index=X.columns).reset_index()
vif_df.columns = ['feature','vif_score']
vif_df = vif_df.loc[vif_df.feature!='const']
```

	feature	vif_score
1	gre_score	4.489983
2	toefl_score	3.664298
3	univ_ranking	2.572110
4	motiv_letter_strength	2.785764
5	recommendation_strength	1.977698
6	gpa	4.654540
7	research_exp	1.518065

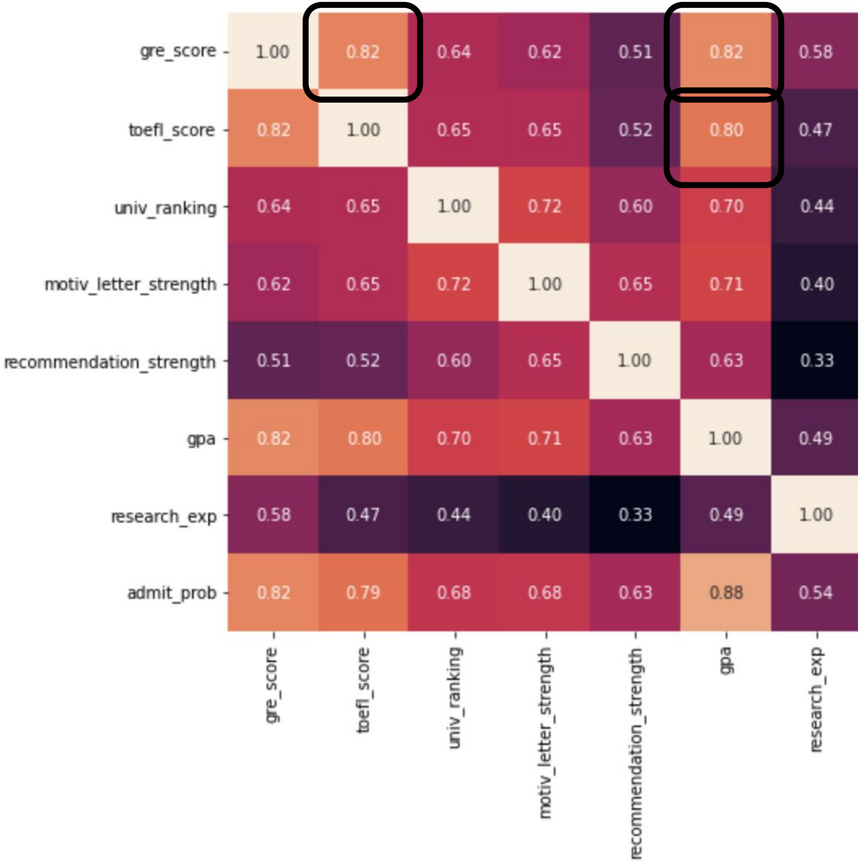
Feature Correlation

To prevent multicollinearity

- We can draw a correlation heatmap
  - Using `sns.heatmap()`

- We found that `gre_score`, `toefl_score`, and `gpa` are highly correlated each other
  - We decide to include only `gpa` to represent these three features
  - Because it is the most correlated with the target variable

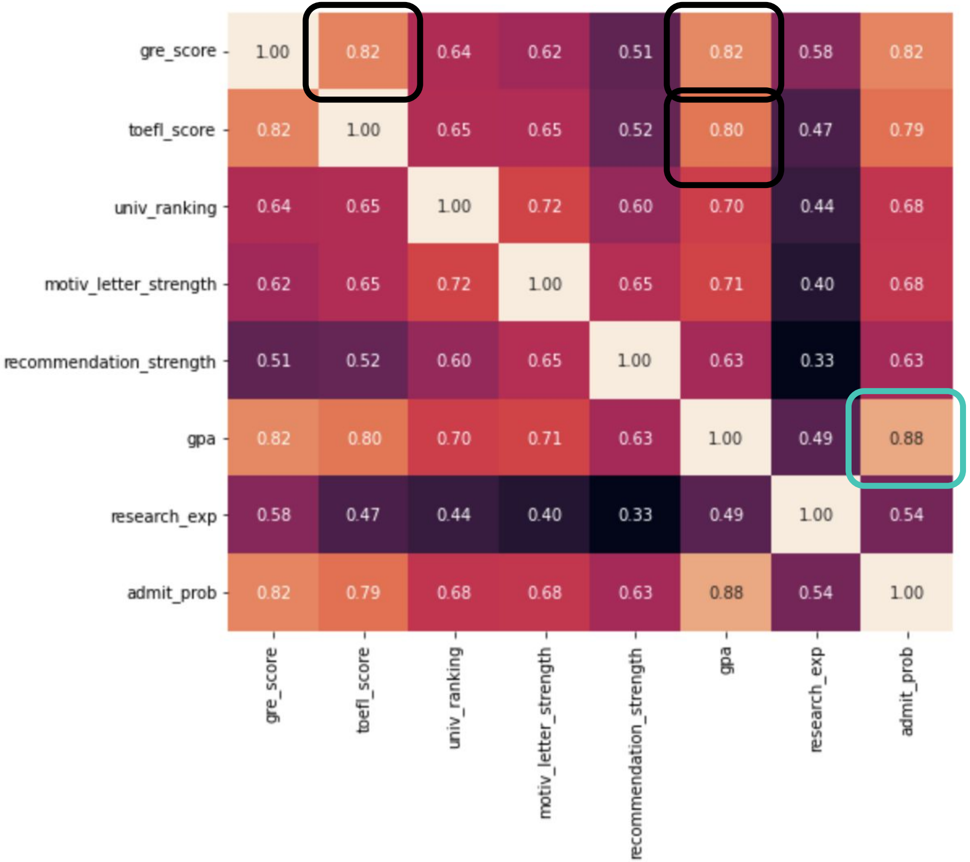
- ***Note: Threshold: abs(corr) >= 0.8***



Feature Correlation

To prevent multicollinearity

- We can draw a correlation heatmap
  - Using `sns.heatmap()`
- We found that `gre_score`, `toefl_score`, and `gpa` are highly correlated each other
  - We decide to include only `gpa` to represent these three features
  - Because it is the most correlated with the target variable
- ***Note: Threshold: abs(corr) >= 0.8***



Training the Model

Excluding `gre_score` and `toefl_score`

```
feature_admit_train = feature_admit_train.drop(columns=['gre_score','toefl_score'])
feature_admit_test = feature_admit_test.drop(columns=['gre_score','toefl_score'])
```

```
from sklearn.linear_model import LinearRegression

# define the model
multi_reg = LinearRegression()

# train the model
X_admit_train = feature_admit_train.to_numpy()
y_admit_train = target_admit_train.to_numpy()

multi_reg.fit(X_admit_train, y_admit_train)
```

Interpreting the Obtained Model

```
coef_df
```

	feature	coefficient
0	intercept	-0.766426
1	univ_ranking	0.006984
2	motiv_letter_strength	0.004346
3	recommendation_strength	0.014776
4	gpa	0.161004
5	research_exp	0.038274

```
admit_prob = -0.763 + 0.007  univ_ranking
             + 0.004 motiv_letter
             + 0.015 recom + 0.161 gpa
             + 0.038 research
```

Sample coeff interpretation:
An increase of 1 point in GPA, while the other features are kept fixed, is associated with an increase of 0.161 point in admit_prob

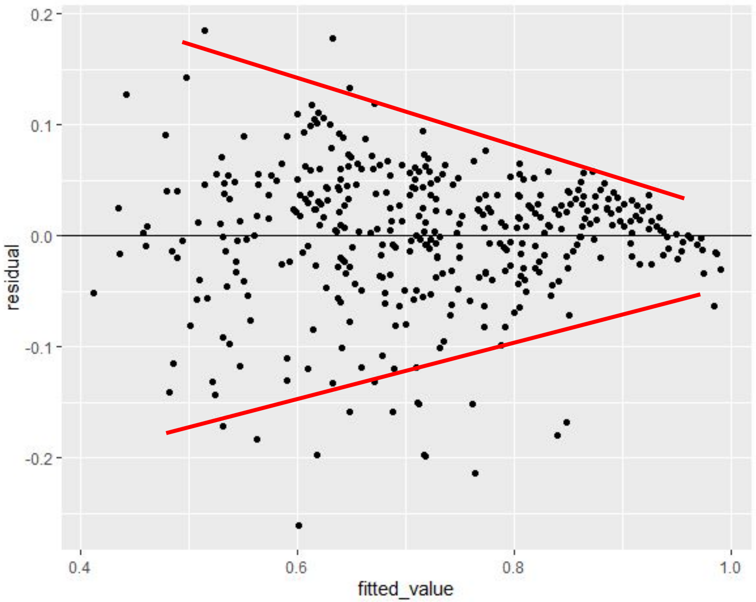
Residual Plot

To check three assumptions of linear regression

- For >1 predictor: residual vs `predicted_target`

```
# prepare dataframe
# >1 predictor --> predicted value VS residual
df_resid = pd.DataFrame({
    'x_axis': y_predict_train,
    'residual': residual
})

# residual plot
sns.scatterplot(data=df_resid, x="x_axis", y="residual")
plt.axhline(0)
plt.show()
```



- Linear relationship ✔️
- Constant variance ❌
- Independent observations ✔️

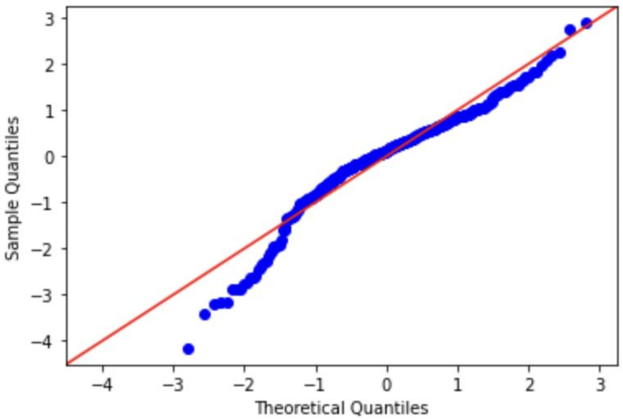
QQ-Plot

The same as before

```
# QQplot
from sklearn.preprocessing import StandardScaler

std_resid = StandardScaler().fit_transform(residual.reshape(-1,1))
std_resid = np.array([value for nested_array in std_resid for value in nested_array])

import statsmodels.api as sm
sm.qqplot(std_resid, line='45')
plt.show()
```



- Normality ✔
  - Slightly skewed, though

R2 Score

- R2 score measures portion of variability of the target variable that is successfully explained (modelled) by the features included in the model

- The higher == the better
  - Max value (not possible, though) is 100%

```
# R^2 score
from sklearn.metrics import r2_score

r2_score(y_admit_train,y_predict_train)
```

```
0.7986824284294713
```

- Interpretation: 79.86% of variability of `admit_prob` is successfully explained using all the features in the model

Hands-On

- Open today's Jupyter notebook on your Google Colab!
- Make sure you have uploaded the required CSV files to your google drive
    - Remember the file path!

Model Evaluation (1/2)

- After we are satisfied (enough) with the model, we perform model evaluation
- Essentially, we check how is the **model performance on test data**
- To do so, we will compute evaluation metric
    - Root Mean Squared Error (RMSE):
        - The standard deviation of our prediction errors with respect to the regression line
        - It is a measure of how spread out the residuals are

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Model Evaluation (2/2)

```
from sklearn.metrics import mean_squared_error

print('RMSE for testing data is {}'.format(np.sqrt(mean_squared_error(y_admit_test, y_predict_test))))
```

```
RMSE for testing data is 0.05880540869550099
```

- RMSE = 0.058
  - The standard deviation of prediction errors is 0.058
  - i.e. from the regression line, the residuals mostly deviate between +- 0.058

Any

Question

Thank you