Data Science II with python (Class notes)

STAT 303-2

Arvind Krishna

1/3/2023

Table of contents

Preface

These are class notes for the course STAT303-2. This is not the course text-book. You are required to read the relevant sections of the book as mentioned on the course website.

The course notes are currently being written, and will continue to being developed as the course progresses (just like the course textbook last quarter). Please report any typos / mistakes / inconsistencies / issues with the class notes / class presentations in your comments here. Thank you!

Part I

Linear regression

1 Simple Linear Regression

Read section 3.1 of the book before using these notes.

Note that in this course, lecture notes are not sufficient, you must read the book for better understanding. Lecture notes are just implementing the concepts of the book on a dataset, but not explaining the concepts elaborately.

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import seaborn as sns
import matplotlib.pyplot as plt
```

Develop a simple linear regression model that predicts car price based on engine size. Datasets to be used: *Car_features_train.csv, Car_prices_train.csv*

```
trainf = pd.read_csv('./Datasets/Car_features_train.csv')
trainp = pd.read_csv('./Datasets/Car_prices_train.csv')
train = pd.merge(trainf,trainp)
train.head()
```

	carID	brand	model	year	transmission	mileage	fuelType	tax	mpg	engineSize	price
0	18473	bmw	6 Series	2020	Semi-Auto	11	Diesel	145	53.3282	3.0	37980
1	15064	bmw	6 Series	2019	Semi-Auto	10813	Diesel	145	53.0430	3.0	33980
2	18268	bmw	6 Series	2020	Semi-Auto	6	Diesel	145	53.4379	3.0	36850
3	18480	bmw	6 Series	2017	Semi-Auto	18895	Diesel	145	51.5140	3.0	25998
4	18492	bmw	6 Series	2015	Automatic	62953	Diesel	160	51.4903	3.0	18990

```
#Using the ols function to create an ols object. 'ols' stands for 'Ordinary least squares'
ols_object = smf.ols(formula = 'price~engineSize', data = train)
```

```
#Using the fit() function of the 'ols' class to fit the model
model = ols_object.fit()
```

```
#Printing model summary which contains among other things, the model coefficients
model.summary()
```

Table 1.2: OLS Regression Results

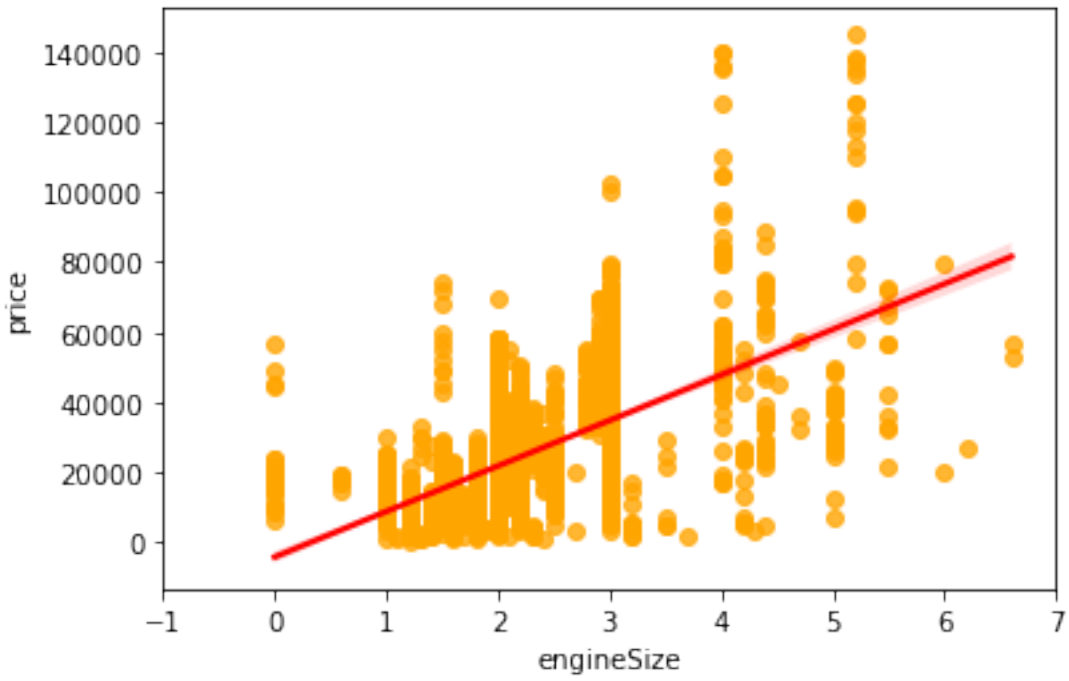
Dep. Variable:	price	R-squared:	0.390
Model:	OLS	Adj. R-squared:	0.390
Method:	Least Squares	F-statistic:	3177.
Date:	Thu, 19 Jan 2023	Prob (F-statistic):	0.00
Time:	16:44:04	Log-Likelihood:	-53949.
No. Observations:	4960	AIC:	1.079e+05
Df Residuals:	4958	BIC:	1.079e+05
Df Model:	1		
Covariance Type:	nonrobust		

The model equation is: car price = -4122.0357 + 12990 * engineSize

Visualize the regression line

```
sns.regplot(x = 'engineSize', y = 'price', data = train, color = 'orange',line_kws={"color
plt.xlim(-1,7)
#Note that some of the engineSize values are 0. They are incorrect, and should ideally be
```

```
(-1.0, 7.0)
```



Predict the car price for the cars in the test dataset. Datasets to be used: *Car_features_test.csv, Car_prices_test.csv*

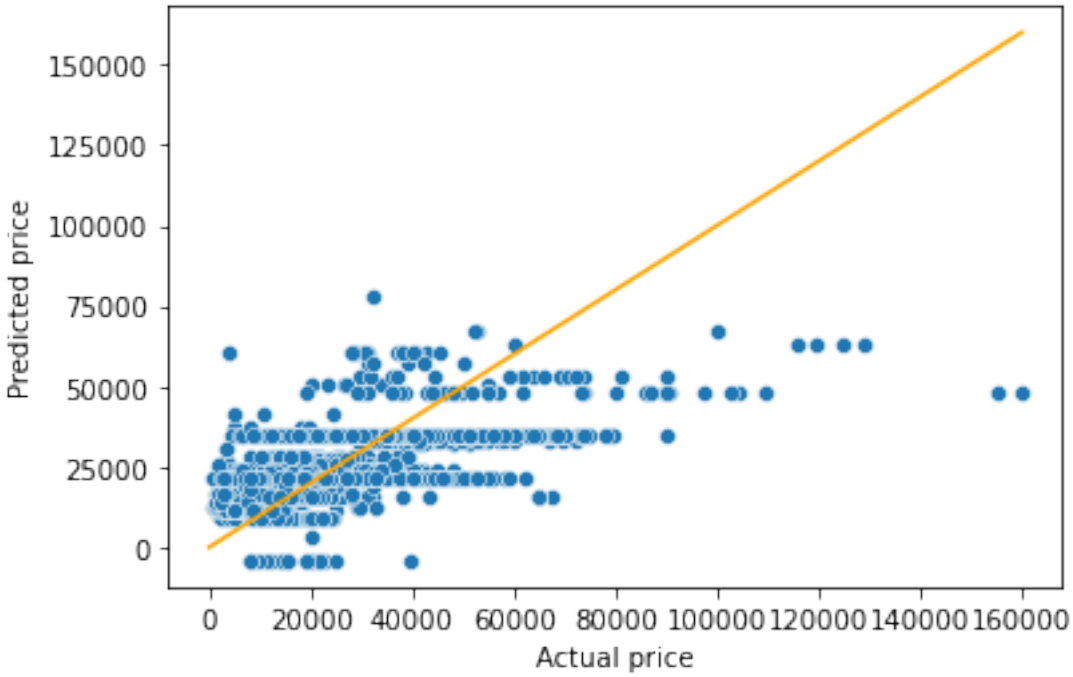
```
testf = pd.read_csv('./Datasets/Car_features_test.csv')
testp = pd.read_csv('./Datasets/Car_prices_test.csv')
```

```
#Using the predict() function associated with the 'model' object to make predictions of ca
pred_price = model.predict(testf)#Note that the predict() function finds the predictor 'en
```

Make a visualization that compares the predicted car prices with the actual car prices

```
sns.scatterplot(x = testp.price, y = pred_price)
#In case of a perfect prediction, all the points must lie on the line x = y.
sns.lineplot(x = [0,testp.price.max()], y = [0,testp.price.max()],color='orange') #Plottin
plt.xlabel('Actual price')
plt.ylabel('Predicted price')
```

```
Text(0, 0.5, 'Predicted price')
```



The prediction doesn't look too good. This is because we are just using one predictor - engine size. We can probably improve the model by adding more predictors when we learn multiple linear regression.

What is the RMSE of the predicted car price?

```
np.sqrt(((testp.price - pred_price)**2).mean())
```

```
12995.1064515487
```

The root mean squared error in predicting car price is around $13k.

What is the residual standard error based on the training data?

```
np.sqrt(model.mse_resid)
```

```
12810.109175214136
```

The residual standard error on the training data is close to the RMSE on the test data. This shows that the performance of the model on unknown data is comparable to its performance

on known data. This implies that the model is not overfitting, which is good! In case we overfit a model on the training data, its performance on unknown data is likely to be worse than that on the training data.

Find the confidence and prediction intervals of the predicted car price

```
#Using the get_prediction() function associated with the 'model' object to get the interva
intervals = model.get_prediction(testf)
```

```
#The function requires specifying alpha (probability of Type 1 error) instead of the confi
intervals.summary_frame(alpha=0.05)
```

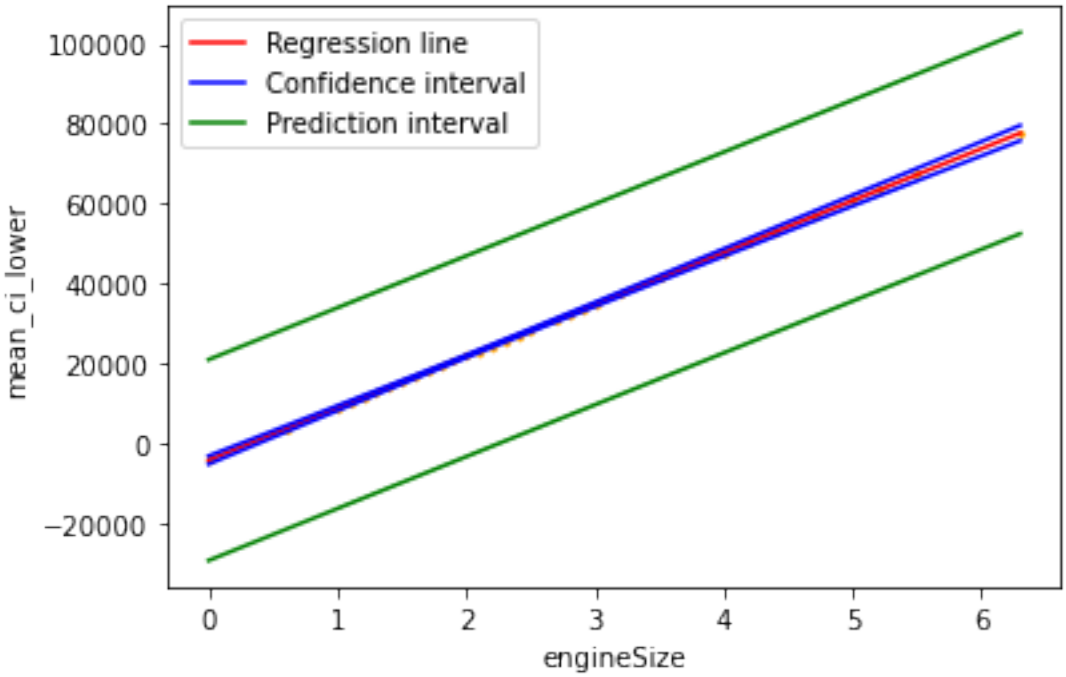
	mean	mean_se	mean_ci_lower	mean_ci_upper	obs_ci_lower	obs_ci_upper
0	34842.807319	271.666459	34310.220826	35375.393812	9723.677232	59961.937406
1	34842.807319	271.666459	34310.220826	35375.393812	9723.677232	59961.937406
2	34842.807319	271.666459	34310.220826	35375.393812	9723.677232	59961.937406
3	8866.245277	316.580850	8245.606701	9486.883853	-16254.905974	33987.396528
4	47831.088340	468.949360	46911.740050	48750.436631	22700.782946	72961.393735
...	...	...	...	...	...	...
2667	47831.088340	468.949360	46911.740050	48750.436631	22700.782946	72961.393735
2668	34842.807319	271.666459	34310.220826	35375.393812	9723.677232	59961.937406
2669	8866.245277	316.580850	8245.606701	9486.883853	-16254.905974	33987.396528
2670	21854.526298	184.135754	21493.538727	22215.513869	-3261.551421	46970.604017
2671	21854.526298	184.135754	21493.538727	22215.513869	-3261.551421	46970.604017

Show the regression line predicting car price based on engine size for test data. Also show the confidence and prediction intervals for the car price.

```
interval_table = intervals.summary_frame(alpha=0.05)
```

```
sns.scatterplot(x = testf.engineSize, y = pred_price,color = 'orange', s = 10)
sns.lineplot(x = testf.engineSize, y = pred_price, color = 'red')
sns.lineplot(x = testf.engineSize, y = interval_table.mean_ci_lower, color = 'blue')
sns.lineplot(x = testf.engineSize, y = interval_table.mean_ci_upper, color = 'blue',label=
sns.lineplot(x = testf.engineSize, y = interval_table.obs_ci_lower, color = 'green')
sns.lineplot(x = testf.engineSize, y = interval_table.obs_ci_upper, color = 'green')
plt.legend(labels=["Regression line","Confidence interval", "Prediction interval"])
```

```
<matplotlib.legend.Legend at 0x26a3a32c550>
```


Regression line
Confidence interval
Prediction interval
100000
80000
60000
40000
20000
0
−20000
mean_ci_lower
0
1
2
3
4
5
6
engineSize

2 Multiple Linear Regression

Read section 3.2 of the book before using these notes.

Note that in this course, lecture notes are not sufficient, you must read the book for better understanding. Lecture notes are just implementing the concepts of the book on a dataset, but not explaining the concepts elaborately.

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import seaborn as sns
import matplotlib.pyplot as plt
```

Develop a multiple linear regression model that predicts car price based on engine size, year, mileage, and mpg. Datasets to be used: *Car_features_train.csv, Car_prices_train.csv*

```
trainf = pd.read_csv('./Datasets/Car_features_train.csv')
trainp = pd.read_csv('./Datasets/Car_prices_train.csv')
train = pd.merge(trainf,trainp)
train.head()
```

	carID	brand	model	year	transmission	mileage	fuelType	tax	mpg	engineSize	price
0	18473	bmw	6 Series	2020	Semi-Auto	11	Diesel	145	53.3282	3.0	37980
1	15064	bmw	6 Series	2019	Semi-Auto	10813	Diesel	145	53.0430	3.0	33980
2	18268	bmw	6 Series	2020	Semi-Auto	6	Diesel	145	53.4379	3.0	36850
3	18480	bmw	6 Series	2017	Semi-Auto	18895	Diesel	145	51.5140	3.0	25998
4	18492	bmw	6 Series	2015	Automatic	62953	Diesel	160	51.4903	3.0	18990

```
#Using the ols function to create an ols object. 'ols' stands for 'Ordinary least squares'
ols_object = smf.ols(formula = 'price~year+mileage+mpg+engineSize', data = train)
model = ols_object.fit()
model.summary()
```

Table 2.2: OLS Regression Results

Dep. Variable:	price	R-squared:	0.660
Model:	OLS	Adj. R-squared:	0.660
Method:	Least Squares	F-statistic:	2410.
Date:	Tue, 27 Dec 2022	Prob (F-statistic):	0.00
Time:	01:07:25	Log-Likelihood:	-52497.
No. Observations:	4960	AIC:	1.050e+05
Df Residuals:	4955	BIC:	1.050e+05
Df Model:	4		
Covariance Type:	nonrobust		

The model equation is: estimated car price = -3.661e6 + 1818 * year -0.15 * mileage - 79.31 * mpg + 12180 * engineSize

Predict the car price for the cars in the test dataset. Datasets to be used: *Car_features_test.csv, Car_prices_test.csv*

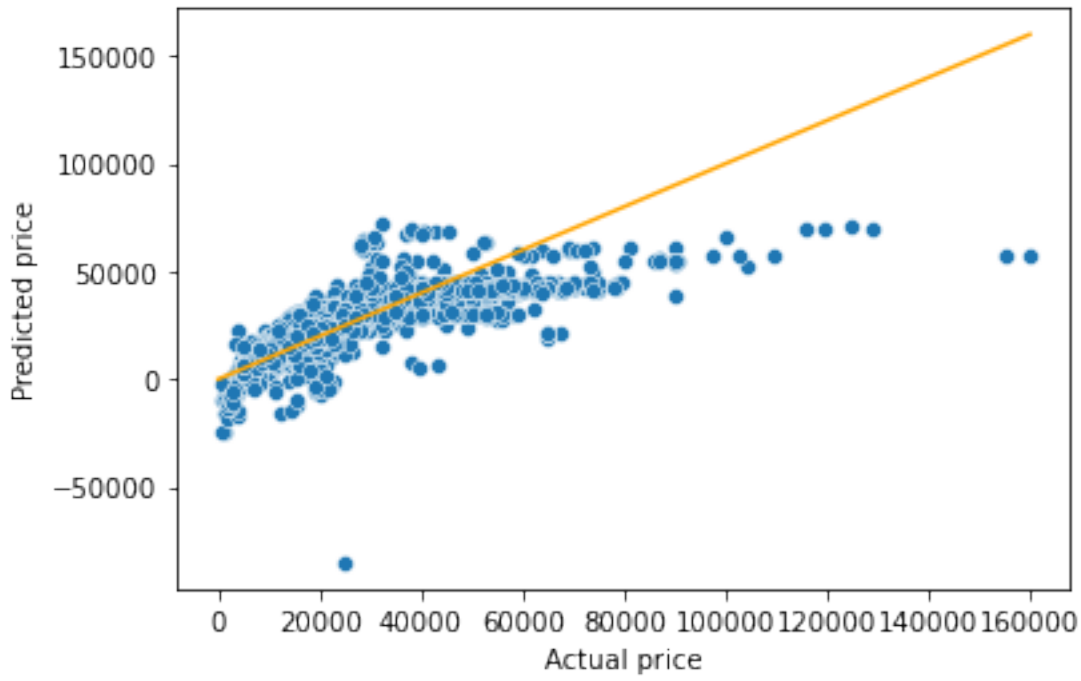
```
testf = pd.read_csv('./Datasets/Car_features_test.csv')
testp = pd.read_csv('./Datasets/Car_prices_test.csv')
```

```
#Using the predict() function associated with the 'model' object to make predictions of ca
pred_price = model.predict(testf)#Note that the predict() function finds the predictor 'en
```

Make a visualization that compares the predicted car prices with the actual car prices

```
sns.scatterplot(x = testp.price, y = pred_price)
#In case of a perfect prediction, all the points must lie on the line x = y.
sns.lineplot(x = [0,testp.price.max()], y = [0,testp.price.max()],color='orange') #Plottin
plt.xlabel('Actual price')
plt.ylabel('Predicted price')
```

```
Text(0, 0.5, 'Predicted price')
```



The prediction looks better as compared to the one with simple linear regression. This is because we have four predictors to help explain the variation in car price, instead of just one in the case of simple linear regression. Also, all the predictors have a significant relationship with price as evident from their p-values. Thus, all four of them are contributing in explaining the variation. Note the higher values of R2 as compared to the one in the case of simple linear regression.

What is the RMSE of the predicted car price?

```
np.sqrt(((testp.price - pred_price)**2).mean())
```

```
9956.82497993548
```

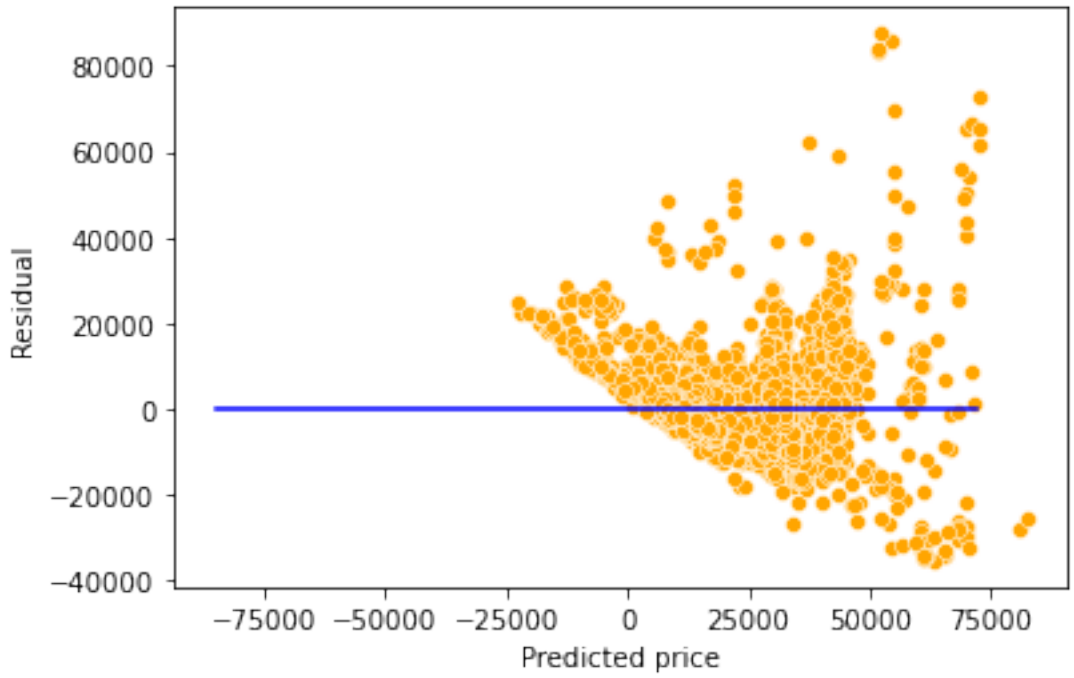
What is the residual standard error based on the training data?

```
np.sqrt(model.mse_resid)
```

```
9563.74782917604
```

```
sns.scatterplot(x = model.fittedvalues, y=model.resid,color = 'orange')
sns.lineplot(x = [pred_price.min(),pred_price.max()],y = [0,0],color = 'blue')
plt.xlabel('Predicted price')
plt.ylabel('Residual')
```

```
Text(0, 0.5, 'Residual')
```



Will the explained variation (R-squared) in car price always increase if we add a variable?

Should we keep on adding variables as long as the explained variation (R-squared) is increasing?

```
#Using the ols function to create an ols object. 'ols' stands for 'Ordinary least squares'
np.random.seed(1)
train['rand_col'] = np.random.rand(train.shape[0])
ols_object = smf.ols(formula = 'price~year+mileage+mpg+engineSize+rand_col', data = train)
model = ols_object.fit()
model.summary()
```

Table 2.3: OLS Regression Results

Dep. Variable:	price	R-squared:	0.661
Model:	OLS	Adj. R-squared:	0.660
Method:	Least Squares	F-statistic:	1928.
Date:	Tue, 27 Dec 2022	Prob (F-statistic):	0.00
Time:	01:07:38	Log-Likelihood:	-52497.
No. Observations:	4960	AIC:	1.050e+05
Df Residuals:	4954	BIC:	1.050e+05
Df Model:	5		
Covariance Type:	nonrobust		

Adding a variable with random values to the model (*rand_col*) increased the explained variation (R-squared). This is because the model has one more parameter to tune to reduce the residual squared error (RSS). However, the p-value of *rand_col* suggests that its coefficient is zero. Thus, using the model with *rand_col* may give poorer performance on unknown data, as compared to the model without *rand_col*. This implies that it is not a good idea to blindly add variables in the model to increase R-squared.

3 Variable interactions and transformations

Read sections 3.3.1 and 3.3.2 of the book before using these notes.

Note that in this course, lecture notes are not sufficient, you must read the book for better understanding. Lecture notes are just implementing the concepts of the book on a dataset, but not explaining the concepts elaborately.

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import seaborn as sns
import matplotlib.pyplot as plt
```

```
trainf = pd.read_csv('./Datasets/Car_features_train.csv')
trainp = pd.read_csv('./Datasets/Car_prices_train.csv')
testf = pd.read_csv('./Datasets/Car_features_test.csv')
testp = pd.read_csv('./Datasets/Car_prices_test.csv')
train = pd.merge(trainf,trainp)
train.head()
```

	carID	brand	model	year	transmission	mileage	fuelType	tax	mpg	engineSize	price
0	18473	bmw	6 Series	2020	Semi-Auto	11	Diesel	145	53.3282	3.0	37980
1	15064	bmw	6 Series	2019	Semi-Auto	10813	Diesel	145	53.0430	3.0	33980
2	18268	bmw	6 Series	2020	Semi-Auto	6	Diesel	145	53.4379	3.0	36850
3	18480	bmw	6 Series	2017	Semi-Auto	18895	Diesel	145	51.5140	3.0	25998
4	18492	bmw	6 Series	2015	Automatic	62953	Diesel	160	51.4903	3.0	18990

Until now, we have assumed that the association between a predictor X_j and response Y does not depend on the value of other predictors. For example, the multiple linear regression model that we developed in Chapter 2 assumes that the average increase in price associated with a unit increase in engineSize is always \$12,180, regardless of the value of other predictors. However, this assumption may be incorrect.

3.0.1 Variable interaction between continuous predictors

We can relax this assumption by considering another predictor, called an interaction term. Let us assume that the average increase in `price` associated with a one-unit increase in `engineSize` depends on the model `year` of the car. In other words, there is an interaction between `engineSize` and `year`. This interaction can be included as a predictor, which is the product of `engineSize` and `year`. *Note that there are several possible interactions that we can consider. Here the interaction between* ***`engineSize`*** *and* ***`year`*** *is just an example.*

```
#Considering interaction between engineSize and year
ols_object = smf.ols(formula = 'price~year*engineSize+mileage+mpg', data = train)
model = ols_object.fit()
model.summary()
```

Table 3.2: OLS Regression Results

Dep. Variable:	price	R-squared:	0.682
Model:	OLS	Adj. R-squared:	0.681
Method:	Least Squares	F-statistic:	2121.
Date:	Tue, 24 Jan 2023	Prob (F-statistic):	0.00
Time:	15:28:11	Log-Likelihood:	-52338.
No. Observations:	4960	AIC:	1.047e+05
Df Residuals:	4954	BIC:	1.047e+05
Df Model:	5		
Covariance Type:	nonrobust		

Note that the R-squared has increased as compared to the model in Chapter 2 since we added a predictor.

The model equation is:

$$price = \beta_0 + \beta_1 * year + \beta_2 * engineSize + \beta_3 * (year * engineSize) + \beta 4 * mileage + \beta_5 * mpg, \quad (3.1)$$

or

$$price = \beta_0 + \beta_1 * year + (\beta_2 + \beta_3 * year) * engineSize + \beta 4 * mileage + \beta_5 * mpg, \quad (3.2)$$

or

$$price = \beta_0 + \beta_1 * year + \tilde{\beta} * engineSize + \beta 4 * mileage + \beta_5 * mpg, \quad (3.3)$$

Since $\tilde{\beta}$ is a function of `year`, the association between `engineSize` and `price` is no longer a constant. A change in the value of `year` will change the association between `price` and `engineSize`.

Substituting the values of the coefficients:

$$price = 5.606e5 - 275.3833 * year + (-1.796e6 + 896.7687 * year) * engineSize - 0.1525 * mileage - 84.3417 * mpg \tag{3.4}$$

Thus, for cars launched in the year 2010, the average increase in price for one liter increase in engine size is -1.796e6 + 896.7687 * 2010 $\approx$ \$6,500, assuming all the other predictors are constant. However, for cars launched in the year 2020, the average increase in price for one liter increase in engine size is -1.796e6 + 896.7687*2020 $\approx$ \$15,500 , assuming all the other predictors are constant.

Similarly, the equation can be re-arranged as:

$$price = 5.606e5 + (-275.3833 + 896.7687 * engineSize) * year - 1.796e6 * engineSize - 0.1525 * mileage - 84.3417 * mp \tag{3.5}$$

Thus, for cars with an engine size of 2 litres, the average increase in price for a one year newer model is -275.3833+896.7687 * 2 $\approx$ \$1500, assuming all the other predictors are constant. However, for cars with an engine size of 3 litres, the average increase in price for a one year newer model is -275.3833+896.7687 * 3 $\approx$ \$2400, assuming all the other predictors are constant.

```
#Computing the RMSE of the model with the interaction term
pred_price = model.predict(testf)
np.sqrt(((testp.price - pred_price)**2).mean())
```

```
9423.598872501092
```

Note that the RMSE is lower than that of the model in Chapter 2. This is because the interaction term between `engineSize` and `year` is significant and relaxes the assumption of constant association between price and engine size, and between price and year. This added flexibility makes the model better fit the data. Caution: Too much flexibility may lead to overfitting!

Note that interaction terms corresponding to other variable pairs, and higher order interaction terms (such as those containing 3 or 4 variables) may also be significant and improve the model fit & thereby the prediction accuracy of the model.

3.0.2 Including qualitative predictors in the model

Let us develop a model for predicting `price` based on `engineSize` and the qualitative predictor `transmission`.

```
#checking the distribution of values of transmission
train.transmission.value_counts()
```

```
Manual       1948
Automatic    1660
Semi-Auto    1351
Other           1
Name: transmission, dtype: int64
```

Note that the *Other* category of the variable *transmission* contains only a single observation, which is likely to be insufficient to train the model. We'll remove that observation from the training data. Another option may be to combine the observation in the *Other* category with the nearest category, and keep it in the data.

```
train_updated = train[train.transmission!='Other']
```

```
ols_object = smf.ols(formula = 'price~engineSize+transmission', data = train_updated)
model = ols_object.fit()
model.summary()
```

Table 3.3: OLS Regression Results

Dep. Variable:	price	R-squared:	0.459
Model:	OLS	Adj. R-squared:	0.458
Method:	Least Squares	F-statistic:	1400.
Date:	Tue, 24 Jan 2023	Prob (F-statistic):	0.00
Time:	15:28:21	Log-Likelihood:	-53644.
No. Observations:	4959	AIC:	1.073e+05
Df Residuals:	4955	BIC:	1.073e+05
Df Model:	3		
Covariance Type:	nonrobust		

Note that there is no coefficient for the *Automatic* level of the variable `Transmission`. If a car doesn't have *Manual* or *Semi-Automatic* transmission, then it has an *Automatic* transmission.

Thus, the coefficient of *Automatic* will be redundant, and the dummy variable corresponding to *Automatic* transmission is dropped from the model.

The level of the categorical variable that is dropped from the model is called the baseline level. Here *Automatic* transmission is the baseline level. The coefficients of other levels of `transmission` should be interpreted with respect to the baseline level.

Q: Interpret the intercept term

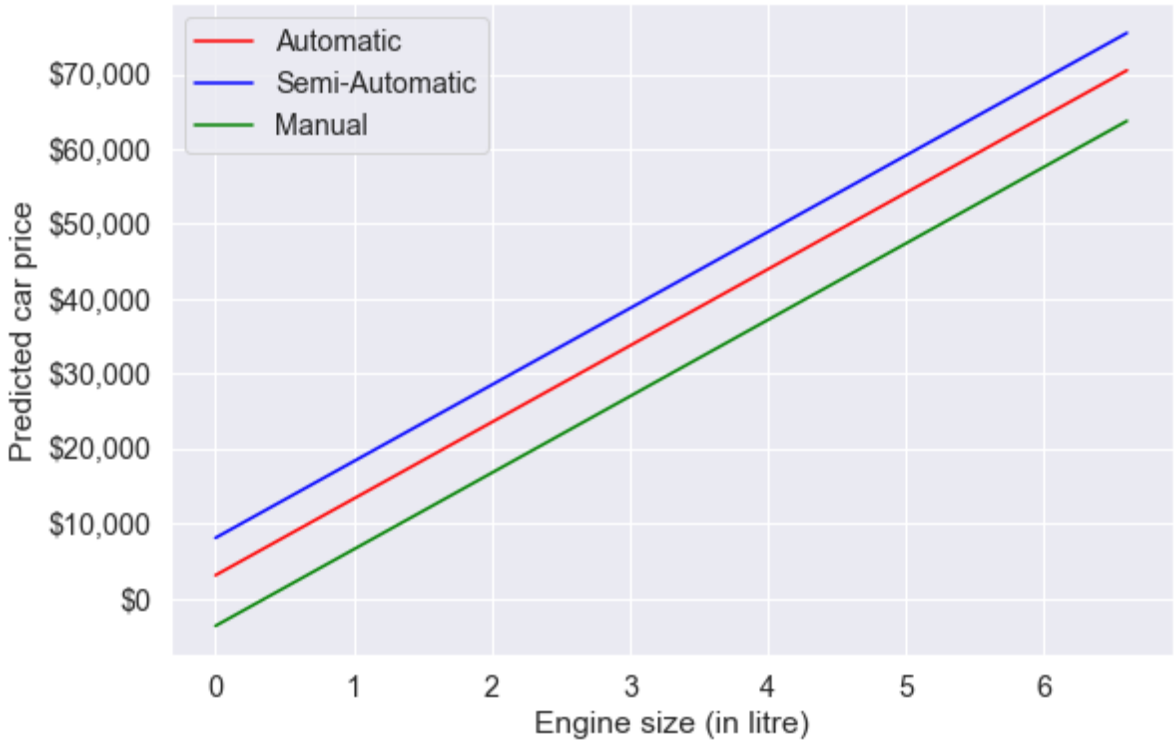
Ans: For the hypothetical scenario of a car with zero engine size and *Automatic* transmission, the estimated mean car price is $\approx$ \$3042.

Q: Interpret the coefficient of `transmission[T.Manual]`

Ans: The estimated mean price of a car with manual transmission is $\approx$ \$6770 less than that of a car with *Automatic* transmission.

Let us visualize the developed model.

```
#Visualizing the developed model
plt.rcParams["figure.figsize"] = (9,6)
sns.set(font_scale = 1.3)
x = np.linspace(train_updated.engineSize.min(),train_updated.engineSize.max(),100)
ax = sns.lineplot(x = x, y = model.params['engineSize']*x+model.params['Intercept'], color
sns.lineplot(x = x, y = model.params['engineSize']*x+model.params['Intercept']+model.param
sns.lineplot(x = x, y = model.params['engineSize']*x+model.params['Intercept']+model.param
plt.legend(labels=["Automatic","Semi-Automatic", "Manual"])
plt.xlabel('Engine size (in litre)')
plt.ylabel('Predicted car price')
ax.yaxis.set_major_formatter('${x:,.0f}')
```



Based on the developed model, for a given engine size, the car with a semi-automatic transmission is estimated to be the most expensive on average, while the car with a manual transmission is estimated to be the least expensive on average.

Changing the baseline level: By default, the baseline level is chosen as the one that comes first if the levels are arranged in alphabetical order. However, you can change the baseline level by specifying one explicitly.

Internally, statsmodels uses the patsy package to convert formulas and data to the matrices that are used in model fitting. You may refer to this section in the patsy documentation to specify a particular level of the categorical variable as the baseline.

For example, suppose we wish to change the baseline level to *Manual* transmission. We can specify this in the formula as follows:

```
ols_object = smf.ols(formula = 'price~engineSize+C(transmission, Treatment("Manual"))', da
model = ols_object.fit()
model.summary()
```

Table 3.4: OLS Regression Results

Dep. Variable:	price	R-squared:	0.459
Model:	OLS	Adj. R-squared:	0.458
Method:	Least Squares	F-statistic:	1400.
Date:	Tue, 24 Jan 2023	Prob (F-statistic):	0.00
Time:	15:28:39	Log-Likelihood:	-53644.
No. Observations:	4959	AIC:	1.073e+05
Df Residuals:	4955	BIC:	1.073e+05
Df Model:	3		
Covariance Type:	nonrobust		

3.0.3 Including qualitative predictors and their interaction with continuous predictors in the model

Note that the qualitative predictor leads to fitting 3 parallel lines to the data, as there are 3 categories.

However, note that we have made the constant association assumption. The fact that the lines are parallel means that the average increase in car price for one litre increase in engine size does not depend on the type of transmission. This represents a potentially serious limitation of the model, since in fact a change in engine size may have a very different association on the price of an automatic car versus a semi-automatic or manual car.

This limitation can be addressed by adding an interaction variable, which is the product of `engineSize` and the dummy variables for semi-automatic and manual transmissions.

```
#Using the ols function to create an ols object. 'ols' stands for 'Ordinary least squares'
ols_object = smf.ols(formula = 'price~engineSize*transmission', data = train_updated)
model = ols_object.fit()
model.summary()
```

Table 3.5: OLS Regression Results

Dep. Variable:	price	R-squared:	0.479
Model:	OLS	Adj. R-squared:	0.478
Method:	Least Squares	F-statistic:	909.9
Date:	Sun, 22 Jan 2023	Prob (F-statistic):	0.00
Time:	22:55:55	Log-Likelihood:	-53550.
No. Observations:	4959	AIC:	1.071e+05
Df Residuals:	4953	BIC:	1.072e+05
Df Model:	5		

Table 3.5: OLS Regression Results

Covariance Type:	nonrobust

The model equation for the model with interactions is:

Automatic transmission: $price = 3754.7238 + 9928.6082 * engineSize$,
Semi-Automatic transmission: $price = 3754.7238 + 9928.6082 * engineSize + (-5282.7164 + 4162.2428 * engineSize)$,
Manual transmission: $price = 3754.7238 + 9928.6082 * engineSize + (1768.5856 - 5285.9059 * engineSize)$, or

Automatic transmission: $price = 3754.7238 + 9928.6082 * engineSize$,
Semi-Automatic transmission: $price = -1527 + 7046 * engineSize$,
Manual transmission: $price = 5523 + 4642 * engineSize$,

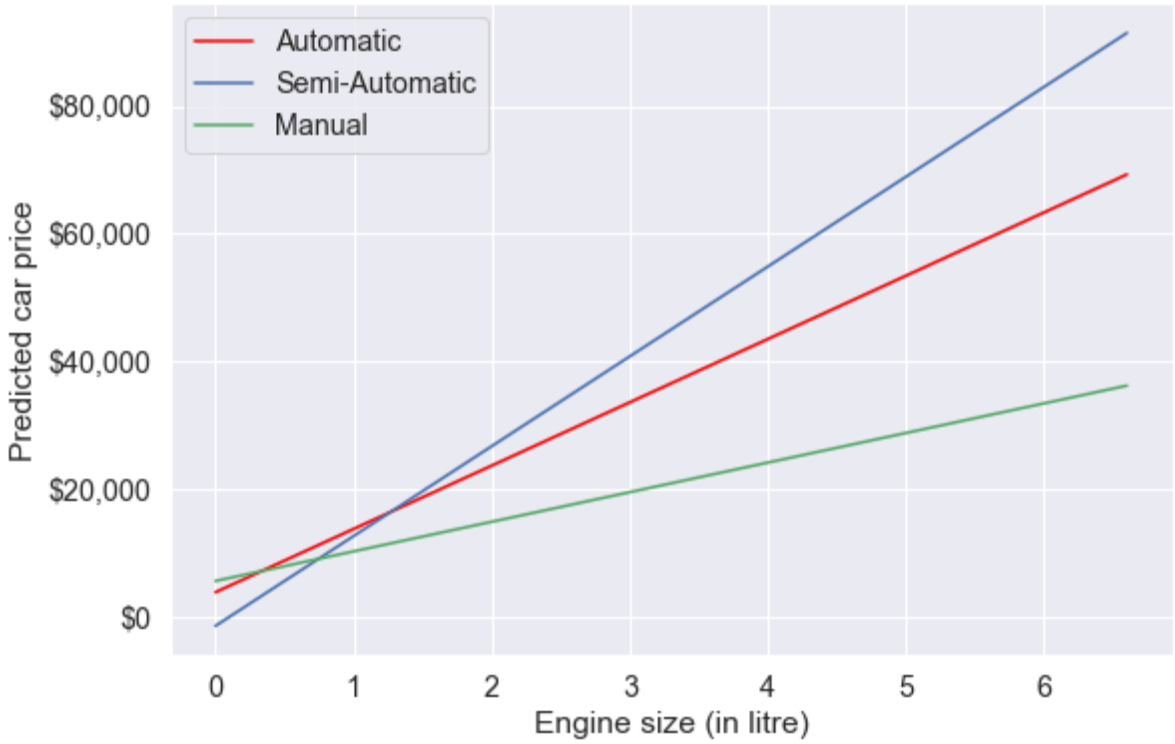
Q: Interpret the coefficient of manual tranmission, i.e., the coefficient of `transmission[T.Manual]`.

A: For a given engine size, the estimated mean `price` of a car with *Manual* transmission is $\approx$ \$1768 more than the estimated mean `price` of a car with *Automatic* transmission.

Q: Interpret the coefficient of the interaction between engine size and manual transmission, i.e., the coefficient of `engineSize:transmission[T.Manual]`.

A: For a unit (or a litre) increase in `engineSize` , the increase in estimated mean `price` of a car with *Manual* transmission is $\approx$ \$5285 less than the increase in estimated mean `price` of a car with *Automatic* transmission.

```
#Visualizing the developed model with interaction terms
plt.rcParams["figure.figsize"] = (9,6)
sns.set(font_scale = 1.3)
x = np.linspace(train_updated.engineSize.min(),train_updated.engineSize.max(),100)
ax = sns.lineplot(x = x, y = model.params['engineSize']*x+model.params['Intercept'], label
plt.plot(x, (model.params['engineSize']+model.params['engineSize:transmission[T.Semi-Auto]
plt.plot(x, (model.params['engineSize']+model.params['engineSize:transmission[T.Manual]'])
plt.legend(loc='upper left')
plt.xlabel('Engine size (in litre)')
plt.ylabel('Predicted car price')
ax.yaxis.set_major_formatter('${x:,.0f}')
```



Note the interaction term adds flexibility to the model.

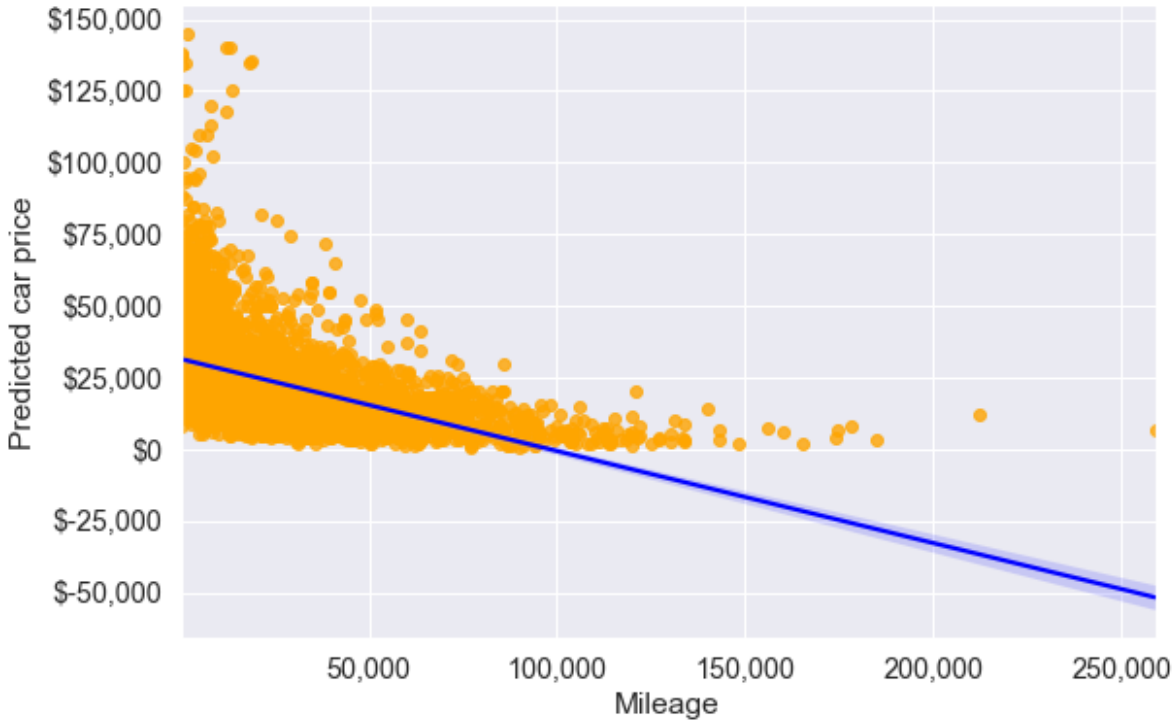
The slope of the regression line for semi-automatic cars is the largest. This suggests that increase in engine size is associated with a higher increase in car price for semi-automatic cars, as compared to other cars.

3.1 Variable transformations

So far we have considered only a linear relationship between the predictors and the response. However, the relationship may be non-linear.

Consider the regression plot of `price` on `mileage`.

```
ax = sns.regplot(x = train_updated.mileage, y =train_updated.price,color = 'orange', line_
plt.xlabel('Mileage')
plt.ylabel('Predicted car price')
ax.yaxis.set_major_formatter('${x:,.0f}')
ax.xaxis.set_major_formatter('{x:,.0f}')
```




```
#R-squared of the model with just mileage
model = smf.ols('price~mileage', data = train_updated).fit()
model.rsquared
```

```
0.22928048993376182
```

From the first scatterplot, we see that the relationship between `price` and `mileage` doesn't seem to be linear, as the points do not lie on a straight line. Also, we see the regression line (or the curve), which is the best fit line doesn't seem to fit the points well. However, `price` on average seems to decrease with `mileage`, albeit in a non-linear manner.

3.1.1 Quadratic transformation

So, we guess that if we model price as a quadratic function of `mileage`, the model may better fit the points (or the curve may better fit the points). Let us transform the predictor `mileage` to include $mileage^2$ (i.e., perform a quadratic transformation on the predictor).

```
#Including mileage squared as a predictor and developing the model
ols_object = smf.ols(formula = 'price~mileage+I(mileage**2)', data = train_updated)
model = ols_object.fit()
model.summary()
```

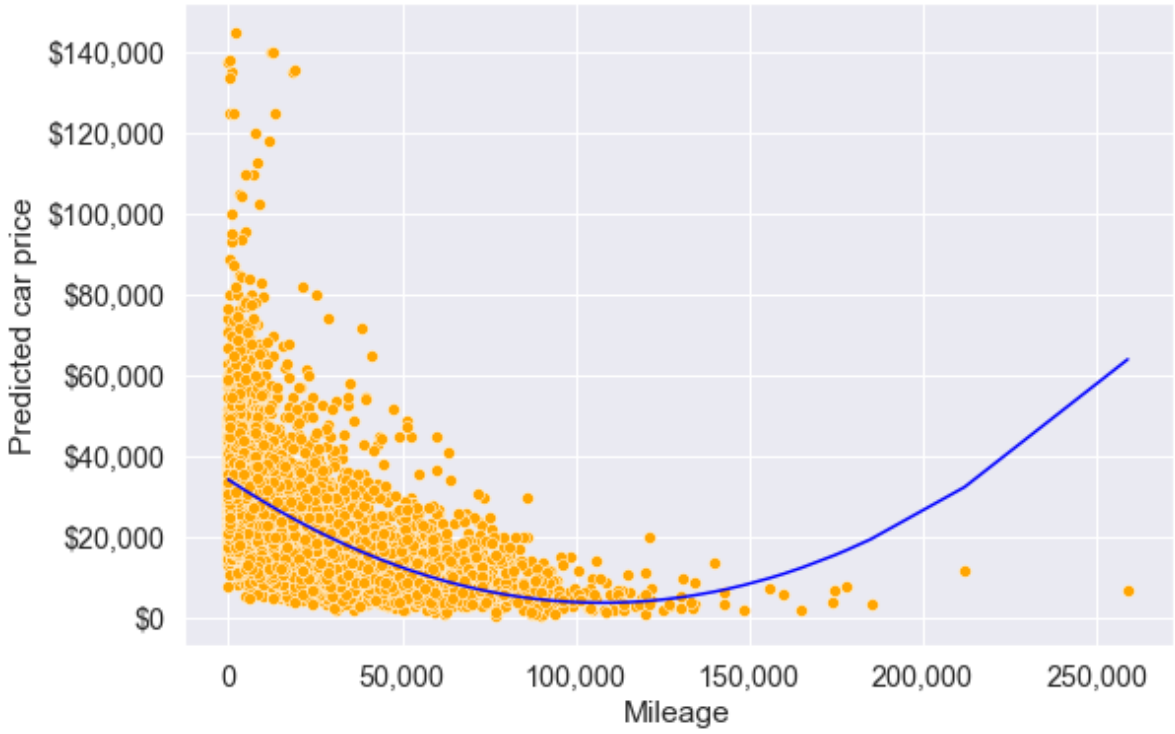
Table 3.6: OLS Regression Results

Dep. Variable:	price	R-squared:	0.271
Model:	OLS	Adj. R-squared:	0.271
Method:	Least Squares	F-statistic:	920.6
Date:	Sun, 22 Jan 2023	Prob (F-statistic):	0.00
Time:	23:26:05	Log-Likelihood:	-54382.
No. Observations:	4959	AIC:	1.088e+05
Df Residuals:	4956	BIC:	1.088e+05
Df Model:	2		
Covariance Type:	nonrobust		

Note that in the formula specified within the `ols()` function, the `I()` operator isolates or insulates the contents within I(…) from the regular formula operators. Without the `I()` operator, `mileage**2` will be treated as the interaction of `mileage` with itself, which is `mileage`. Thus, to add the square of `mileage` as a separate predictor, we need to use the `I()` operator.

Let us visualize the model fit with the quadratic transformation of the predictor - `mileage`.

```
#Visualizing the regression line with the model consisting of the quadratic transformation
pred_price = model.predict(train_updated)
ax = sns.scatterplot(x = 'mileage', y = 'price', data = train_updated, color = 'orange')
sns.lineplot(x = train_updated.mileage, y = pred_price, color = 'blue')
plt.xlabel('Mileage')
plt.ylabel('Predicted car price')
ax.yaxis.set_major_formatter('${x:,.0f}')
ax.xaxis.set_major_formatter('{x:,.0f}')
```



The above model seems to better fit the data (as compared to the model without transformation) at least upto mileage around 125,000. The R^2 of the model with the quadratic transformation of `mileage` is also higher than that of the model without transformation indicating a better fit.

3.1.2 Cubic transformation

Let us see if a cubic transformation of `mileage` can further improve the model fit.

```
#Including mileage squared and mileage cube as predictors and developing the model
ols_object = smf.ols(formula = 'price~mileage+I(mileage**2)+I(mileage**3)', data = train_u
model = ols_object.fit()
model.summary()
```

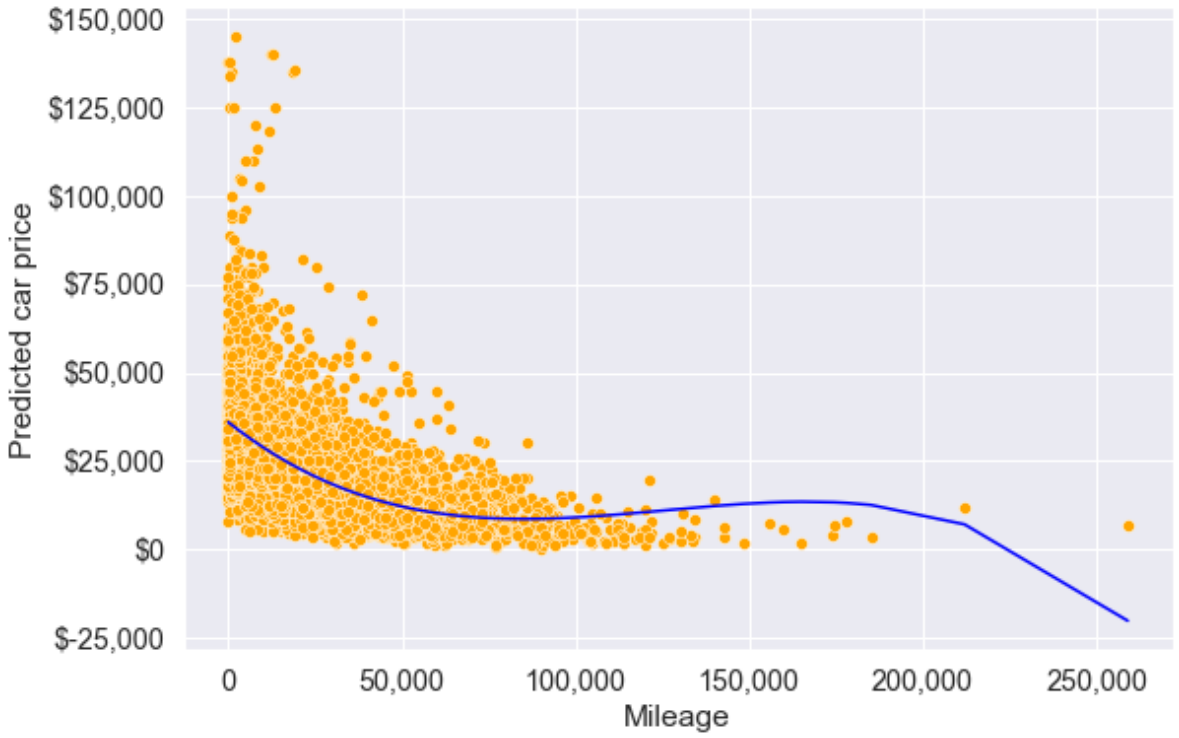
Table 3.7: OLS Regression Results

Dep. Variable:	price	R-squared:	0.283
Model:	OLS	Adj. R-squared:	0.283
Method:	Least Squares	F-statistic:	652.3

Table 3.7: OLS Regression Results

Date:	Sun, 22 Jan 2023	Prob (F-statistic):	0.00
Time:	23:33:27	Log-Likelihood:	-54340.
No. Observations:	4959	AIC:	1.087e+05
Df Residuals:	4955	BIC:	1.087e+05
Df Model:	3		
Covariance Type:	nonrobust		

```
#Visualizing the model with the cubic transformation of mileage
pred_price = model.predict(train_updated)
ax = sns.scatterplot(x = 'mileage', y = 'price', data = train_updated, color = 'orange')
sns.lineplot(x = train_updated.mileage, y = pred_price, color = 'blue')
plt.xlabel('Mileage')
plt.ylabel('Predicted car price')
ax.yaxis.set_major_formatter('${x:,.0f}')
ax.xaxis.set_major_formatter('{x:,.0f}')
```



Note that the model fit with the cubic transformation of `mileage` seems slightly better as

compared to the models with the quadratic transformation, and no transformation of mileage, for mileage up to 180k. However, the model should not be used to predict car prices of cars with a mileage higher than 180k.

Let's update the model created earlier (in the beginning of this chapter) to include the transformed predictor.

```
#Model with an interaction term and a variable transformation term
ols_object = smf.ols(formula = 'price~year*engineSize+mileage+mpg+I(mileage**2)', data = t
model = ols_object.fit()
model.summary()
```

Table 3.8: OLS Regression Results

Dep. Variable:	price	R-squared:	0.702
Model:	OLS	Adj. R-squared:	0.702
Method:	Least Squares	F-statistic:	1947.
Date:	Sun, 22 Jan 2023	Prob (F-statistic):	0.00
Time:	23:42:13	Log-Likelihood:	-52162.
No. Observations:	4959	AIC:	1.043e+05
Df Residuals:	4952	BIC:	1.044e+05
Df Model:	6		
Covariance Type:	nonrobust		

Note that the R-squared has increased as compared to the model with just the interaction term.

```
#Computing RMSE on test data
pred_price = model.predict(testf)
np.sqrt(((testp.price - pred_price)**2).mean())
```

```
9074.494088619422
```

Note that the prediction accuracy of the model has further increased, as the RMSE has reduced. The transformed predictor is statistically significant and provides additional flexibility to better capture the trend in the data, leading to an increase in prediction accuracy.

4 Model assumptions

Read section 3.3.3 (1 & 3) of the book before using these notes.

Note that in this course, lecture notes are not sufficient, you must read the book for better understanding. Lecture notes are just implementing the concepts of the book on a dataset, but not explaining the concepts elaborately.

Consider the model with interactions and transformation developed previously.

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import seaborn as sns
import matplotlib.pyplot as plt
```

```
trainf = pd.read_csv('./Datasets/Car_features_train.csv')
trainp = pd.read_csv('./Datasets/Car_prices_train.csv')
testf = pd.read_csv('./Datasets/Car_features_test.csv')
testp = pd.read_csv('./Datasets/Car_prices_test.csv')
train = pd.merge(trainf,trainp)
train.head()
```

	carID	brand	model	year	transmission	mileage	fuelType	tax	mpg	engineSize	price
0	18473	bmw	6 Series	2020	Semi-Auto	11	Diesel	145	53.3282	3.0	37980
1	15064	bmw	6 Series	2019	Semi-Auto	10813	Diesel	145	53.0430	3.0	33980
2	18268	bmw	6 Series	2020	Semi-Auto	6	Diesel	145	53.4379	3.0	36850
3	18480	bmw	6 Series	2017	Semi-Auto	18895	Diesel	145	51.5140	3.0	25998
4	18492	bmw	6 Series	2015	Automatic	62953	Diesel	160	51.4903	3.0	18990

```
ols_object = smf.ols(formula = 'price~(year+engineSize+mileage+mpg)**2+I(mileage**2)', dat
model = ols_object.fit()
model.summary()
```

Table 4.2: OLS Regression Results

Dep. Variable:	price	R-squared:	0.732
Model:	OLS	Adj. R-squared:	0.731
Method:	Least Squares	F-statistic:	1229.
Date:	Wed, 25 Jan 2023	Prob (F-statistic):	0.00
Time:	11:36:00	Log-Likelihood:	-51911.
No. Observations:	4960	AIC:	1.038e+05
Df Residuals:	4948	BIC:	1.039e+05
Df Model:	11		
Covariance Type:	nonrobust		

```
np.sqrt(model.mse_resid)
```

```
8502.851955843495
```

```
pred_price = model.predict(testf)
np.sqrt(((testp.price - pred_price)**2).mean())
```

```
8708.676318160937
```

```
#Computing MAE on test data
pred_price = model.predict(testf)
(np.abs(testp.price - pred_price)).mean()
```

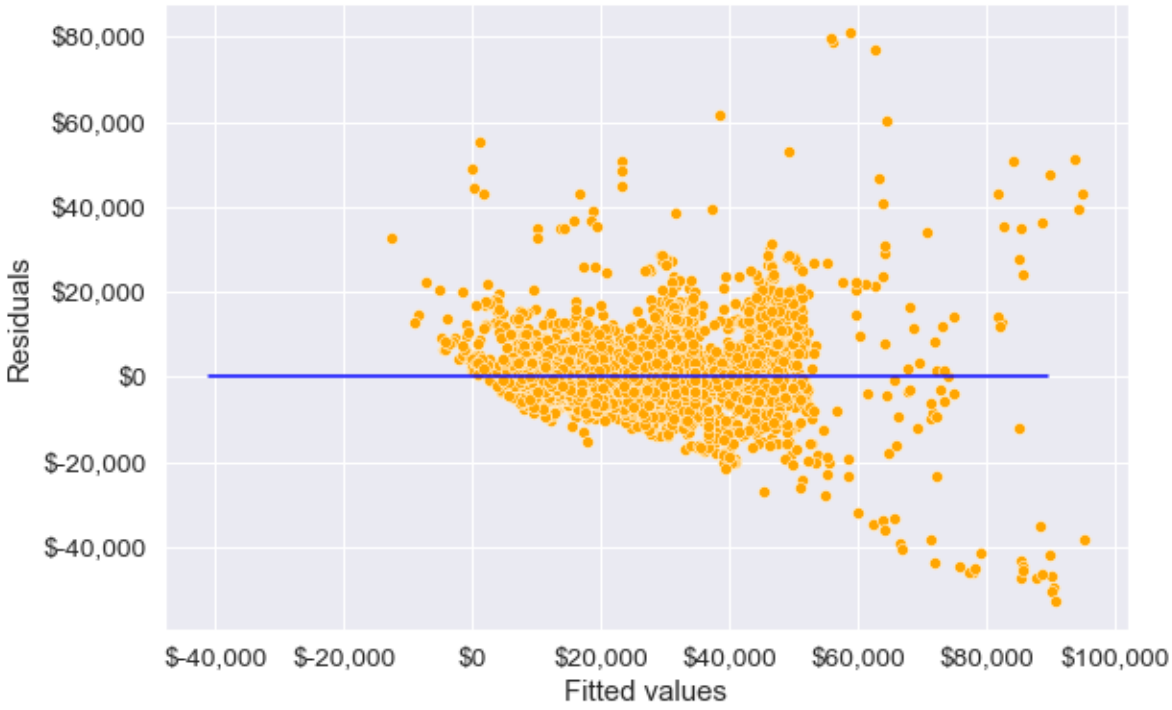
```
5395.006622253402
```

Let us check if this model satisfies the assumptions of the linear regression model

4.1 Non-linearity of data

We have assumed that there is a linear relationship between the predictors and the response. Residual plots, which are scatter plots of residuals vs fitted values, can be used to identify non-linearity. Fitted values are the values estimated by the model on training data, denoted by $\widehat{y}_i$, and residuals are given by $e_i = y_i - \widehat{y}_i$.

```
#Plotting residuals vs fitted values
plt.rcParams["figure.figsize"] = (9,6)
sns.set(font_scale=1.25)
ax = sns.scatterplot(x = model.fittedvalues, y=model.resid,color = 'orange')
sns.lineplot(x = [pred_price.min(),pred_price.max()],y = [0,0],color = 'blue')
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
ax.yaxis.set_major_formatter('${x:,.0f}')
ax.xaxis.set_major_formatter('${x:,.0f}')
```



The model seems to satisfy this assumption, as we do not observe a strong pattern in the residuals around the line Residuals = 0. Residuals are distributed more or less in a similar manner on both sides of the blue line for all fitted values.

For the model to satisfy the linearity assumption perfectly, the points above the line *(Residuals = 0)*, should be mirror image of the points below the line, i.e., the blue line in the above plot should act as a mirror.

What to do if there is non-linear association (page 94 of book): If the residual plot indicates that there are non-linear associations in the data, then a simple approach is to use

non-linear transformations of the predictors, such as $logX, \sqrt{X}$, and X^2, in the regression model.

4.2 Non-constant variance of error terms

The variance of the error terms is assumed to be constant, i.e., $Var(\epsilon_i) = \sigma^2$, and this assumption is used while deriving the standard errors of the regression coefficients. The standard errors in turn are used to test the significant of the predictors, and obtain their confidence interval. Thus, violation of this assumption may lead to incorrect inference. Non-constant variance of error terms, or violation of the constant variance assumption, is called *heteroscedasticity.*

This assumption can be checked by plotting the residuals against fitted values.

```
#Plotting residuals vs fitted values
ax = sns.scatterplot(x = model.fittedvalues, y=model.resid,color = 'orange')
sns.lineplot(x = [pred_price.min(),pred_price.max()],y = [0,0],color = 'blue')
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
ax.yaxis.set_major_formatter('${x:,.0f}')
ax.xaxis.set_major_formatter('${x:,.0f}')
```

We see that the variance of errors seems to increase with increase in the fitted values. In such a case a log transformation of the response can resolve the issue to some extent. This is because a log transformation will result in a higher shrinkage of larger values.

```
#Model with an interaction term and a variable transformation term
ols_object = smf.ols(formula = 'np.log(price)~(year+engineSize+mileage+mpg)**2+I(mileage**
model_log = ols_object.fit()
model_log.summary()
```

Table 4.3: OLS Regression Results

Dep. Variable:	np.log(price)	R-squared:	0.803
Model:	OLS	Adj. R-squared:	0.803
Method:	Least Squares	F-statistic:	1834.
Date:	Wed, 25 Jan 2023	Prob (F-statistic):	0.00
Time:	11:37:55	Log-Likelihood:	-1173.8
No. Observations:	4960	AIC:	2372.
Df Residuals:	4948	BIC:	2450.
Df Model:	11		
Covariance Type:	nonrobust		

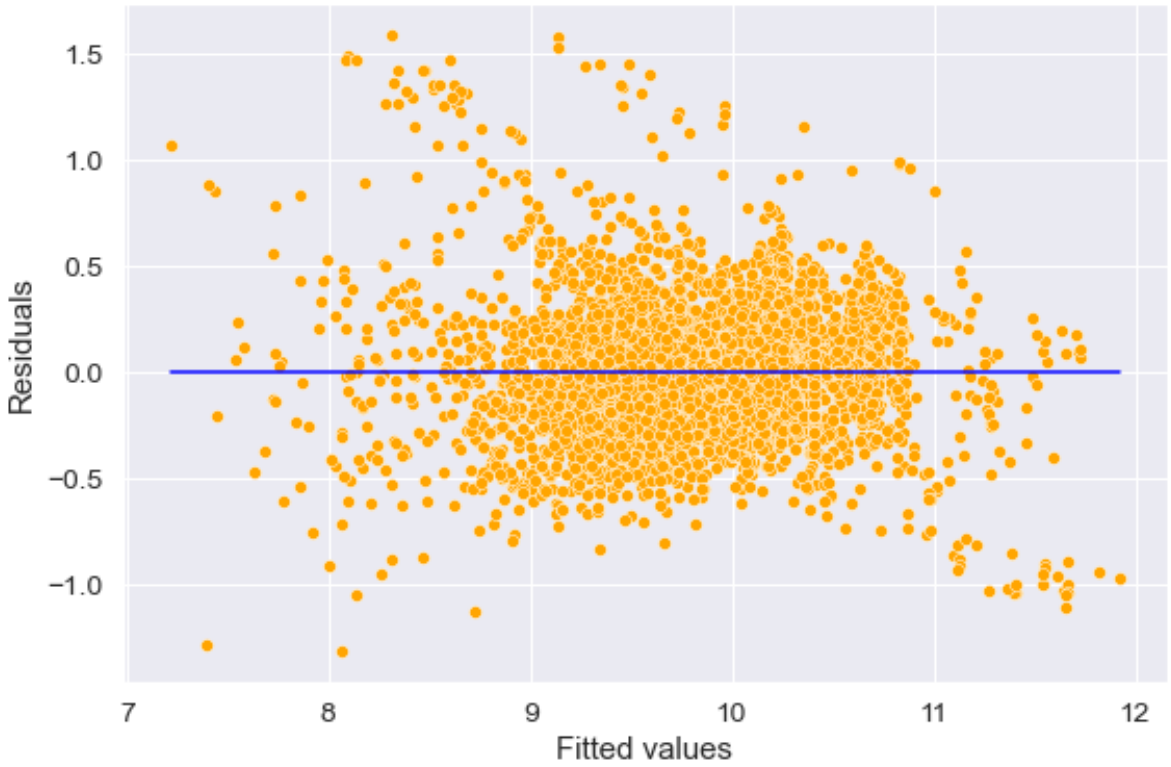
Note that the coefficient of *year* turns out to be significant (at 5% significance level), unlike in the previous model. Intuitively, the coefficient of *year* should have been significant, as *year* has the highest linear correlation of 50% with car *price*.

Although the R-squared has increased as compared to the previous model, violation of this assumption does not cause bias in the regression coefficients. Thus, there may not be a large improvement in the model fit, unless we add predictor(s) to address heteroscedasticity.

Let us check the constant variance assumption again.

```
#Plotting residuals vs fitted values
sns.scatterplot(x = (model_log.fittedvalues), y=(model_log.resid),color = 'orange')
sns.lineplot(x = [model_log.fittedvalues.min(),model_log.fittedvalues.max()],y = [0,0],col
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
```

```
Text(0, 0.5, 'Residuals')
```



Now we observe that the constant variance assumption is satisfied. Let us see the RMSE of this model on test data.

```
#Computing RMSE on test data
pred_price_log = model_log.predict(testf)
np.sqrt(((testp.price - np.exp(pred_price_log))**2).mean())
```

```
9094.209503063496
```

Note that the RMSE of the log-transformed model has increased as compared to the model without transformation. Does it mean the log-transformed model is less accurate?

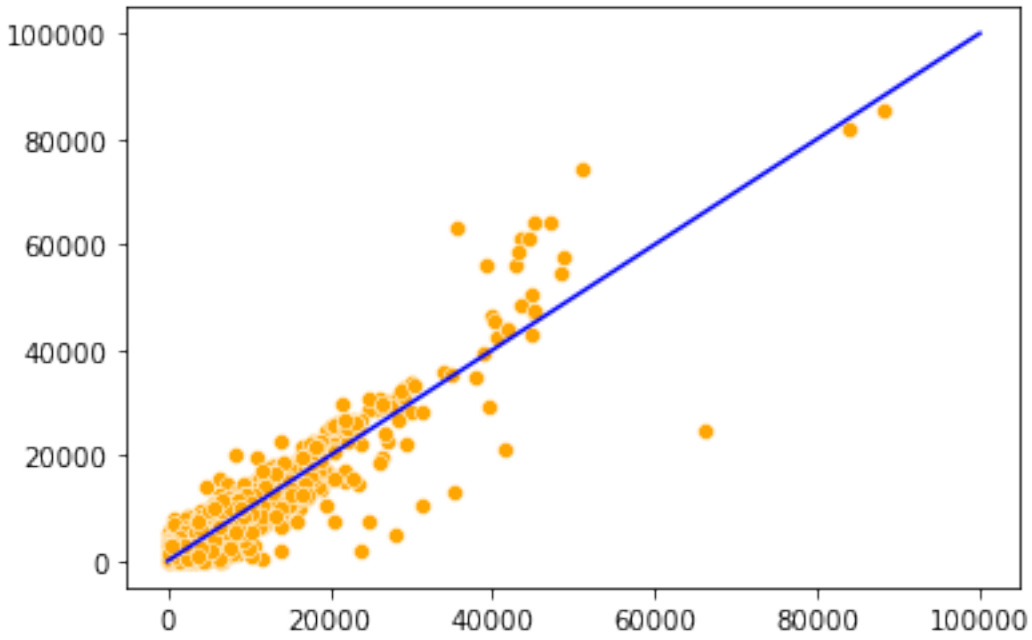
```
#Computing MAE on test data
pred_price_log = model_log.predict(testf)
((np.abs(testp.price - np.exp(pred_price_log))).mean())
```

```
5268.398904745121
```

Although the RMSE has increased a bit for the log-transformed model, the MAE has reduced. This means the log-transformed model does a bit worse on reducing relatively large errors, but does better in reducing the absolute errors on an average.

```
#Comparing errors of the log-transformed model with the previous model
err = np.abs(testp.price - pred_price)
err_log = np.abs(testp.price - np.exp(pred_price_log))
sns.scatterplot(x = err,y = err_log, color = 'orange')
sns.lineplot(x = [0,100000], y = [0,100000], color = 'blue')
np.sum(err_log<err)/len(err)
```

```
0.5572604790419161
```

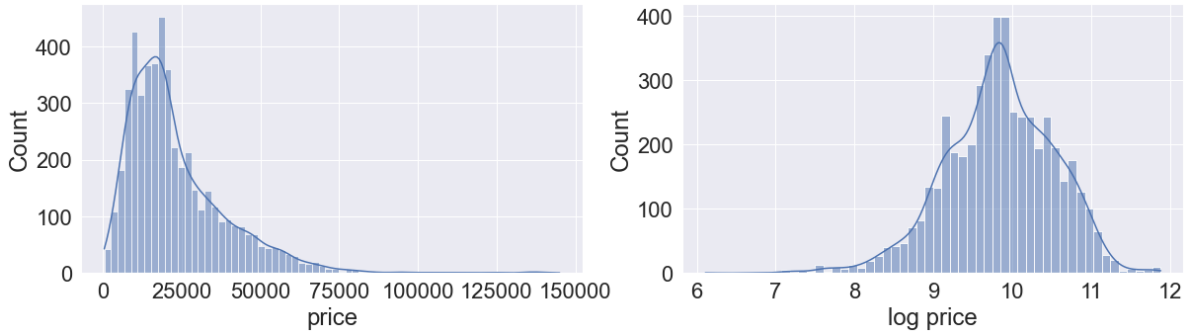


For 56% of the cars, the log transformed makes a more accurate prediction than the previous model, which is another criterion based on which the log-transformed model is more accurate. However, the conclusion based on RMSE is different. This is because RMSE can be influenced by a few large errors. Thus, RMSE, though sometimes appropriate than other criteria, should not be used as the sole measure to compare the accuracy of models.

```
#Visualizing the distribution of price and log(price)
fig = plt.figure()
```

```
fig.subplots_adjust(hspace=0.4, wspace=0.2)
sns.set(rc = {'figure.figsize':(20,12)})
sns.set(font_scale = 2)
ax = fig.add_subplot(2, 2, 1)
sns.histplot(train.price,kde=True)
ax.set(xlabel='price', ylabel='Count')
ax = fig.add_subplot(2, 2, 2)
sns.histplot(np.log(train.price),kde=True)
ax.set(xlabel='log price', ylabel='Count')
```

```
[Text(0.5, 0, 'log price'), Text(0, 0.5, 'Count')]
```



We can see that the log transformation shrinked the higher values of price, making its distribution closer to normal.

Note that heteroscedasticity can also occur due to model misspecification, i.e., in case of missing predictor(s). Some of the cars are too expensive, which makes the `price` distribution skewed. Perhaps, the price of expensive cars could be better explained by the car `model`, a predictor that is missing in the current model.

5 Potential issues

Read section 3.3.3 (4, 5, & 6) of the book before using these notes.

Note that in this course, lecture notes are not sufficient, you must read the book for better understanding. Lecture notes are just implementing the concepts of the book on a dataset, but not explaining the concepts elaborately.

Let us continue with the car price prediction example from the previous chapter.

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

```
trainf = pd.read_csv('./Datasets/Car_features_train.csv')
trainp = pd.read_csv('./Datasets/Car_prices_train.csv')
testf = pd.read_csv('./Datasets/Car_features_test.csv')
testp = pd.read_csv('./Datasets/Car_prices_test.csv')
train = pd.merge(trainf,trainp)
train.head()
```

	carID	brand	model	year	transmission	mileage	fuelType	tax	mpg	engineSize	price
0	18473	bmw	6 Series	2020	Semi-Auto	11	Diesel	145	53.3282	3.0	37980
1	15064	bmw	6 Series	2019	Semi-Auto	10813	Diesel	145	53.0430	3.0	33980
2	18268	bmw	6 Series	2020	Semi-Auto	6	Diesel	145	53.4379	3.0	36850
3	18480	bmw	6 Series	2017	Semi-Auto	18895	Diesel	145	51.5140	3.0	25998
4	18492	bmw	6 Series	2015	Automatic	62953	Diesel	160	51.4903	3.0	18990

```
# Model with an interaction term and a variable transformation term
ols_object = smf.ols(formula = 'np.log(price)~(year+engineSize+mileage+mpg)**2+I(mileage**
model_log = ols_object.fit()
model_log.summary()
```

Table 5.2: OLS Regression Results

Dep. Variable:	np.log(price)	R-squared:	0.803
Model:	OLS	Adj. R-squared:	0.803
Method:	Least Squares	F-statistic:	1834.
Date:	Sun, 05 Feb 2023	Prob (F-statistic):	0.00
Time:	19:31:46	Log-Likelihood:	-1173.8
No. Observations:	4960	AIC:	2372.
Df Residuals:	4948	BIC:	2450.
Df Model:	11		
Covariance Type:	nonrobust		

5.1 Outliers

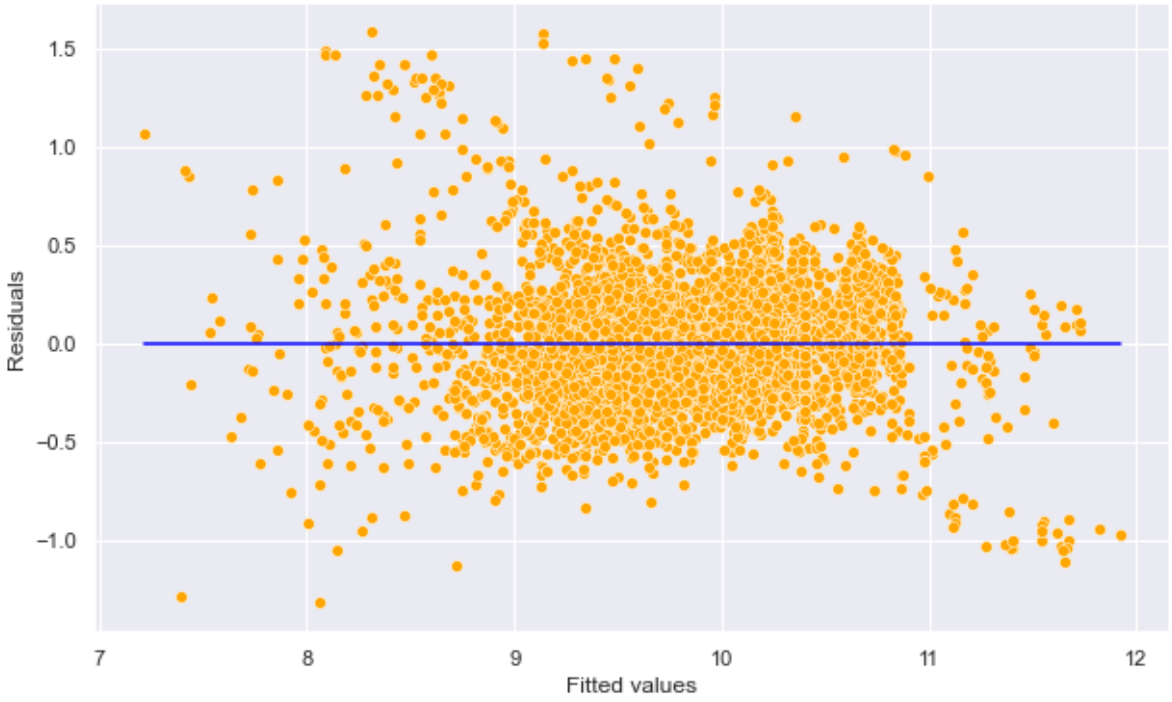
An outlier is a point for which the true response (y_i) is far from the value predicted by the model. Residual plots can be used to identify outliers.

If the the response at the i^{th} observation is y_i, the prediction is $\hat{y}_i$, then the residual e_i is:

$$e_i = y_i - \hat{y}_i$$

```
#Plotting residuals vs fitted values
sns.set(rc={'figure.figsize':(10,6)})
sns.scatterplot(x = (model_log.fittedvalues), y=(model_log.resid),color = 'orange')
sns.lineplot(x = [model_log.fittedvalues.min(),model_log.fittedvalues.max()],y = [0,0],col
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
```

```
Text(0, 0.5, 'Residuals')
```



Some of the errors may be high. However, it is difficult to decide how large a residual needs to be before we can consider a point to be an outlier. To address this problem, we have standardized residuals, which are defined as:

$$r_i = \frac{e_i}{RSE(\sqrt{1-h_{ii}})},$$

where r_i is the standardized residual, RSE is the residual standard error, and h_{ii} is the leverage *(introduced in the next section)* of the i^{th} observation.

Standardized residuals, allow the residuals to be compared on a *standard scale*.

Issue with standardized residuals:, If the observation corresponding to the standardized residual has a high leverage, then it will drag the regression line / plane / hyperplane towards it, thereby influencing the estimate of the residual itself.

Studentized residuals: To address the issue with standardized residuals, studentized residual for the i^{th} observation is computed as the standardized residual, but with the RSE (residual standard error) computed after removing the i^{th} observation from the data. Studentized residual, t_i for the i^{th} observation is given as:

$$t_i = \frac{e_i}{RSE_i(\sqrt{1-h_{ii}})},$$

where RSE_i is the residual standard error of the model developed on the data without the i^{th} observation.

Studentized residuals follow a t distribution with (n–p–2) degrees of freedom. Thus, in general, observations whose studentized residuals have a magnitude higher than 3 are potential outliers.

Let us find the studentized residuals in our car `price` prediction model.

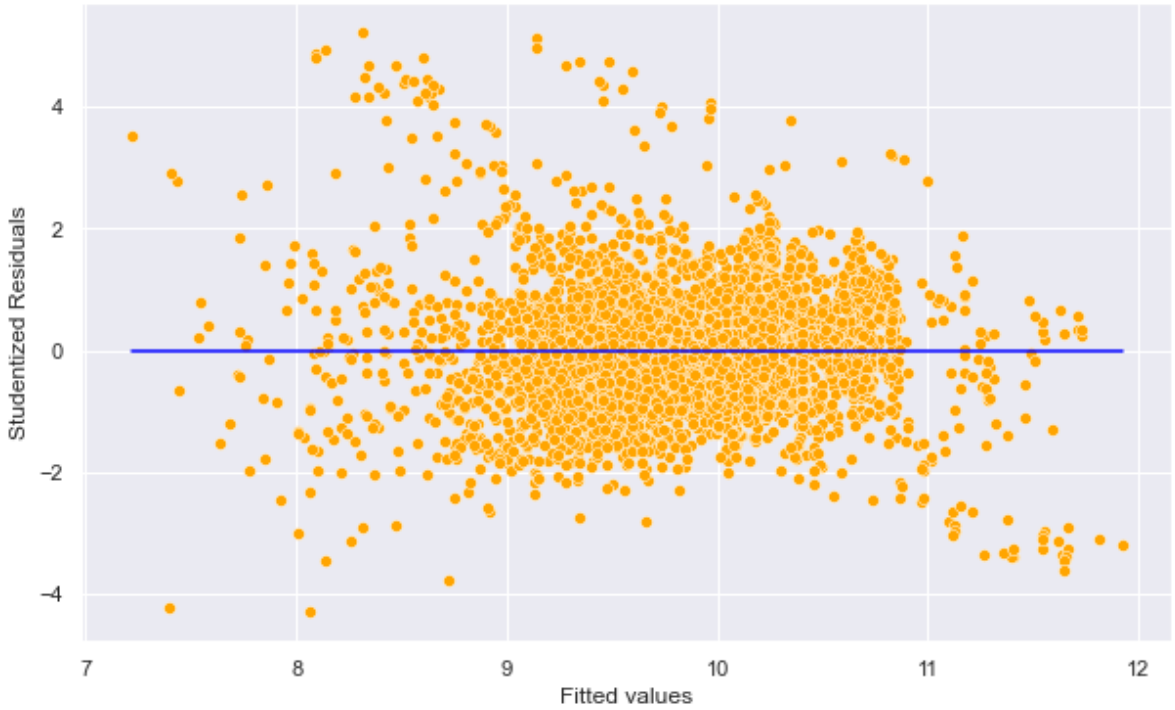
```
#Studentized residuals
out = model_log.outlier_test()
out
```

	student_resid	unadj_p	bonf(p)
0	-1.164204	0.244398	1.0
1	-0.801879	0.422661	1.0
2	-1.263820	0.206354	1.0
3	-0.614171	0.539130	1.0
4	0.027930	0.977719	1.0
...	...	...	...
4955	-0.523361	0.600747	1.0
4956	-0.509539	0.610397	1.0
4957	-1.718802	0.085713	1.0
4958	-0.077595	0.938153	1.0
4959	-0.482388	0.629551	1.0

Studentized residuals are in the first column of the above table.

```
#Plotting studentized residuals vs fitted values
sns.scatterplot(x = (model_log.fittedvalues), y=(out.student_resid),color = 'orange')
sns.lineplot(x = [model_log.fittedvalues.min(),model_log.fittedvalues.max()],y = [0,0],col
plt.xlabel('Fitted values')
plt.ylabel('Studentized Residuals')
```

```
Text(0, 0.5, 'Studentized Residuals')
```



Potential outliers: Observations whose studentized residuals have a magnitude greater than 3.

Impact of outliers: Outliers do not have a large impact on the OLS line / plane / hyperplane. However, outliers do inflate the residual standard error (RSE). RSE in turn is used to compute the standard errors of regression coefficients. As a result, statistically significant variables may appear to be insignificant, and R^2 may appear to be lower.

```
#Number of points with absolute studentized residuals greater than 3
np.sum((np.abs(out.student_resid)>3))
```

```
86
```

Are there outliers in our example?: In the above plot, there are 86 points with absolute studentized residuals larger than 3. However, most of the predictors are significant and R-squared has a relatively high value of 80%. Thus, even if there are outliers, there is no need to remove them as it is unlikely to change the significance of individual variables. Furthermore, looking into the data, we find that the price of some of the luxury cars such as Mercedez G-class is actually much higher than average. So, the potential outliers in the data do not seem to be due to incorrect data. The high studentized residuals may be due to some deficiency in

the model, such as missing predictor(s) (like car `model`), rather than incorrect data. Thus, we should not remove any data that has an outlying value of *log(price)*.

Since `model` seems to be a variable that can explain the price of overly expensive cars, let us include it in the regression model.

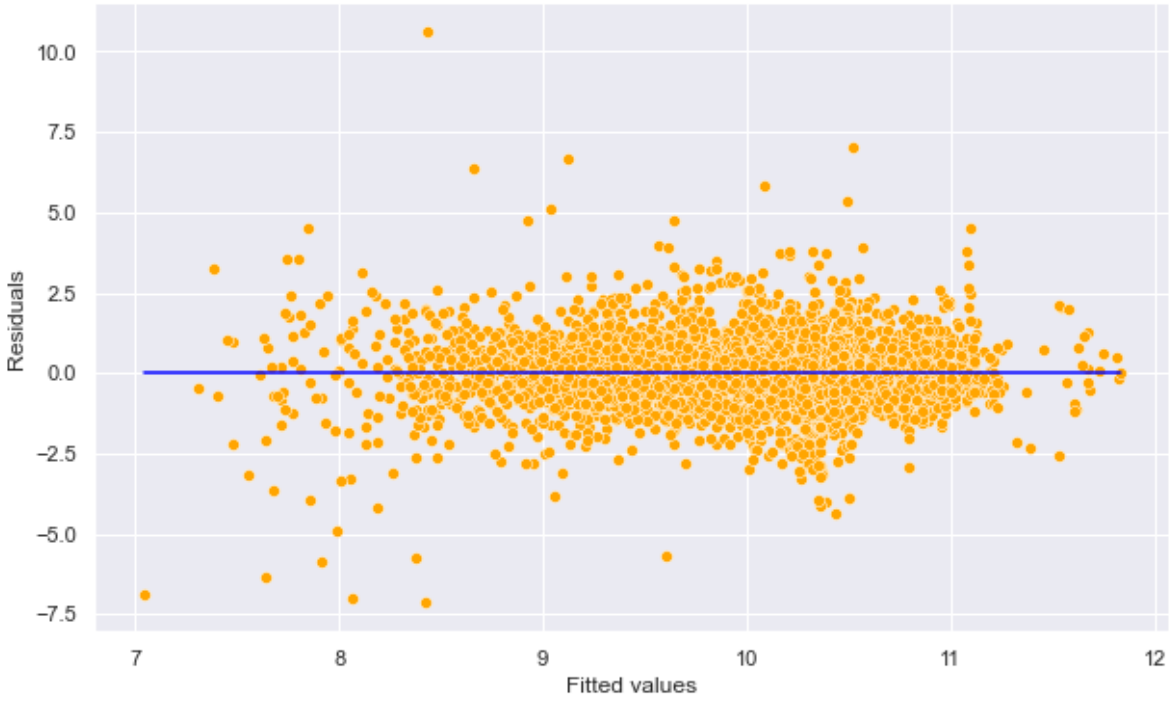
```
#Model with an interaction term and a variable transformation term
ols_object = smf.ols(formula = 'np.log(price)~(year+engineSize+mileage+mpg)**2+I(mileage**
model_log = ols_object.fit()
#Model summary not printed to save space
#model_log.summary()
```

```
#Computing RMSE on test data with car 'model' as one of the predictors
pred_price_log2 = model_log.predict(testf)
np.sqrt(((testp.price - np.exp(pred_price_log2))**2).mean())
```

```
4252.20045604376
```

```
#Plotting studentized residuals vs fitted values for the model with car 'model' as one of
out = model_log.outlier_test()
sns.scatterplot(x = (model_log.fittedvalues), y=(out.student_resid),color = 'orange')
sns.lineplot(x = [model_log.fittedvalues.min(),model_log.fittedvalues.max()],y = [0,0],col
plt.xlabel('Fitted values')
plt.ylabel('Residuals')
```

```
Text(0, 0.5, 'Residuals')
```




```
#Number of points with absolute studentized residuals greater than 3
np.sum((np.abs(out.student_resid)>3))
```

```
69
```

Note the RMSE has reduced to almost half of its value as compared to the regression model without the predictor - `model`. Car model does help better explain the variation in price of cars! The number of points with absolute studentized residuals greater than 3 has also reduced to 69 from 86.

5.2 High leverage points

High leverage points are those with an unsual value of the predictor(s). They have a relatively higher impact on the OLS line / plane / hyperplane, as compared to the outliers.

Leverage statistic (page 99 of the book): In order to quantify an observation's leverage, we compute the leverage statistic. A large value of this statistic indicates an observation with

high leverage. For simple linear regression,

$$h_i = \frac{1}{n} + \frac{(x_i - \bar{x})^2}{\sum_{i'=1}^{n}(x_{i'} - \bar{x})^2}. \tag{5.1}$$

It is clear from this equation that h_i increases with the distance of x_i from $\bar{x}$.The leverage statistic h_i is always between $1/n$ and 1, and the average leverage for all the observations is always equal to $(p+1)/n$. So if a given observation has a leverage statistic that greatly exceeds $(p+1)/n$, then we may suspect that the corresponding point has high leverage.

Influential points: Note that if a high leverage point falls in line with the regression line, then it will not affect the regression line. However, it may inflate R-squared and increase the significance of predictors. If a high leverage point falls away from the regression line, then it is also an outlier, and will affect the regression line. The points whose presence significantly affects the regression line are called influential points. A point that is both a high leverage point and an outlier is likely to be an influential point. However, a high leverage point is not necessarily an influential point.

Source for influential points: https://online.stat.psu.edu/stat501/book/export/html/973

Let us see if there are any high leverage points in our regression model without the predictor - `model`.

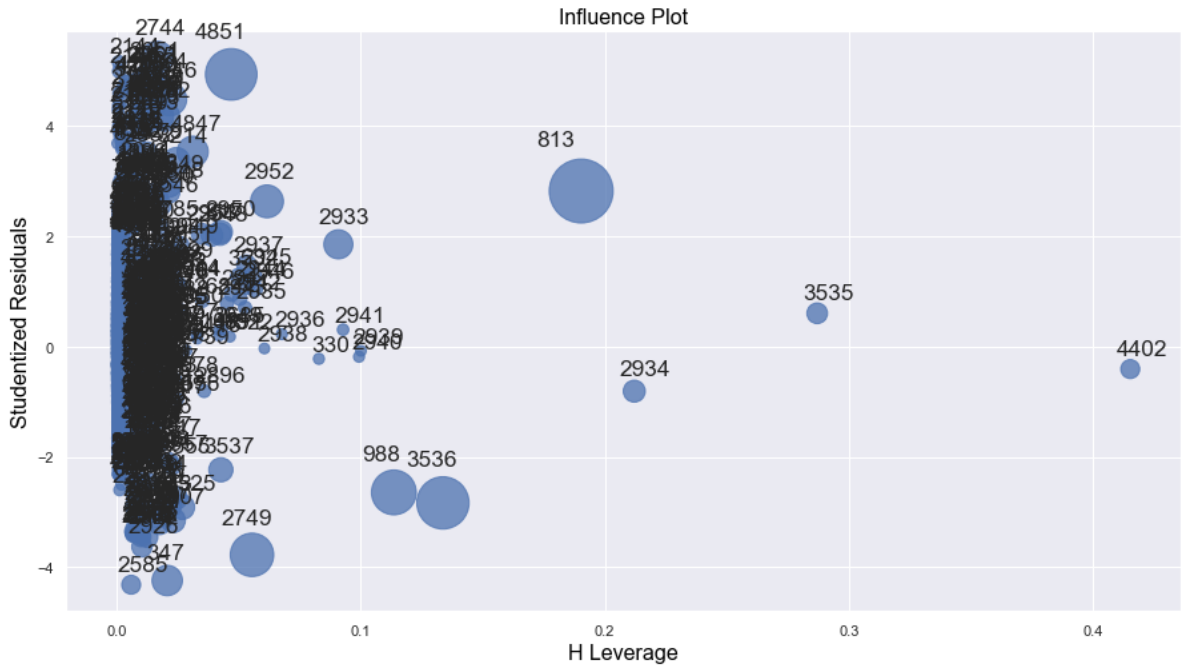
```
#Model with an interaction term and a variable transformation term
ols_object = smf.ols(formula = 'np.log(price)~(year+engineSize+mileage+mpg)**2+I(mileage**
model_log = ols_object.fit()
model_log.summary()
```

Table 5.4: OLS Regression Results

Dep. Variable:	np.log(price)	R-squared:	0.803
Model:	OLS	Adj. R-squared:	0.803
Method:	Least Squares	F-statistic:	1834.
Date:	Sun, 05 Feb 2023	Prob (F-statistic):	0.00
Time:	19:31:59	Log-Likelihood:	-1173.8
No. Observations:	4960	AIC:	2372.
Df Residuals:	4948	BIC:	2450.
Df Model:	11		
Covariance Type:	nonrobust		

```
#Computing the leverage statistic for each observation
influence = model_log.get_influence()
leverage = influence.hat_matrix_diag
```

```
#Visualizng leverage against studentized residuals
sns.set(rc={'figure.figsize':(15,8)})
sm.graphics.influence_plot(model_log);
```



Let us identify the high leverage points in the data, as they may be affecting the regression line if they are outliers as well, i.e., if they are influential points. Note that there is no defined threshold for a point to be classified as a high leverage point. Some statisticians consider points having twice the average leverage as high leverage points, some consider points having thrice the average leverage as high leverage points, and so on.

```
out = model_log.outlier_test()
```

```
#Average leverage of points
average_leverage = (model_log.df_model+1)/model_log.nobs
average_leverage
```

```
0.0024193548387096775
```

Let us consider points having four times the average leverage as high leverage points.

```
#We will remove all observations that have leverage higher than the threshold value.
high_leverage_threshold = 4*average_leverage
```

```
#Number of high leverage points in the dataset
np.sum(leverage>high_leverage_threshold)
```

```
197
```

5.3 Influential points

Observations that are both high leverage points and outliers are influential points that may affect the regression line. Let's remove these influential points from the data and see if it improves the model prediction accuracy on test data.

```
#Dropping influential points from data
train_filtered = train.drop(np.intersect1d(np.where(np.abs(out.student_resid)>3)[0],
                                           (np.where(leverage>high_leverage_threshold)[0])
```

```
train_filtered.shape
```

```
(4921, 11)
```

```
#Number of points removed as they were influential
train.shape[0]-train_filtered.shape[0]
```

```
39
```

We removed 39 influential data points from the training data.

```
#Model after removing the influential observations
ols_object = smf.ols(formula = 'np.log(price)~(year+engineSize+mileage+mpg)**2+I(mileage**
model_log = ols_object.fit()
model_log.summary()
```

Table 5.5: OLS Regression Results

Dep. Variable:	np.log(price)	R-squared:	0.830
Model:	OLS	Adj. R-squared:	0.829
Method:	Least Squares	F-statistic:	2173.
Date:	Sun, 29 Jan 2023	Prob (F-statistic):	0.00
Time:	01:26:25	Log-Likelihood:	-775.51
No. Observations:	4921	AIC:	1575.
Df Residuals:	4909	BIC:	1653.
Df Model:	11		
Covariance Type:	nonrobust		

Note that we obtain a higher R-sqauared value of 83% as compared to 80% with the complete data. Removing the influential points helped obtain a better model fit. However, that may also happen just by reducing observations.

```
#Computing RMSE on test data
pred_price_log = model_log.predict(testf)
np.sqrt(((testp.price - np.exp(pred_price_log))**2).mean())
```

```
8820.685844070766
```

The RMSE on test data has also reduced. This shows that some of the influential points were impacting the regression line. With those points removed, the model better captures the general trend in the data.

5.4 Collinearity

Collinearity refers to the situation when two or more predictor variables have a high linear association. Linear association between a pair of variables can be measured by the correlation coefficient. Thus the correlation matrix can indicate some potential collinearity problems.

5.4.1 Why and how is collinearity a problem

(Source: page 100-101 of book)

The presence of collinearity can pose problems in the regression context, since it can be difficult to separate out the individual effects of collinear variables on the response.

Since collinearity reduces the accuracy of the estimates of the regression coefficients, it causes the standard error for $\hat{\beta}_j$ to grow. Recall that the t-statistic for each predictor is calculated by dividing $\hat{\beta}_j$ by its standard error. Consequently, collinearity results in a decline in the t-statistic. As a result, **in the presence of collinearity, we may fail to reject $H_0 : \beta_j = 0$. This means that the power of the hypothesis test—the probability of correctly detecting a non-zero coefficient—is reduced by collinearity.**

5.4.2 How to measure collinearity/multicollinearity

(Source: page 102 of book)

Unfortunately, not all collinearity problems can be detected by inspection of the correlation matrix: it is possible for collinearity to exist between three or more variables even if no pair of variables has a particularly high correlation. We call this situation multicollinearity. Instead of inspecting the correlation matrix, a better way to assess multicollinearity is to compute the variance inflation factor (VIF). The VIF is variance inflation factor the ratio of the variance of $\hat{\beta}_j$ when fitting the full model divided by the variance of $\hat{\beta}_j$ if fit on its own. The smallest possible value for VIF is 1, which indicates the complete absence of collinearity. Typically in practice there is a small amount of collinearity among the predictors. As a rule of thumb, a **VIF value that exceeds 5 or 10 indicates a problematic amount of collinearity.**

The estimated variance of the coefficient β_j, of the j^{th} predictor X_j, can be expressed as:

$$\hat{var}(\hat{\beta}_j) = \frac{(\hat{\sigma})^2}{(n-1)\hat{var}(X_j)} \cdot \frac{1}{1 - R^2_{X_j|X_{-j}}},$$

where $R^2_{X_j|X_{-j}}$ is the R-squared for the regression of X_j on the other covariates (a regression that does not involve the response variable Y).

In case of simple linear regression, the variance expression in the equation above does not contain the term $\frac{1}{1-R^2_{X_j|X_{-j}}}$, as there is only one predictor. However, in case of multiple linear regression, the variance of the estimate of the j^{th} coefficient ($\hat{\beta}_j$) gets inflated by a factor of $\frac{1}{1-R^2_{X_j|X_{-j}}}$ *(Note that in the complete absence of collinearity, $R^2_{X_j|X_{-j}} = 0$, and the value of this factor will be 1).*

Thus, the Variance inflation factor, or the VIF for the estimated coefficient of the j^{th} predictor X_j is:

$$VIF(\hat{\beta}_j) = \frac{1}{1 - R^2_{X_j|X_{-j}}} \tag{5.2}$$

```
#Correlation matrix
train.corr()
```

	carID	year	mileage	tax	mpg	engineSize	price
carID	1.000000	0.006251	-0.001320	0.023806	-0.010774	0.011365	0.012129
year	0.006251	1.000000	-0.768058	-0.205902	-0.057093	0.014623	0.501296
mileage	-0.001320	-0.768058	1.000000	0.133744	0.125376	-0.006459	-0.478705
tax	0.023806	-0.205902	0.133744	1.000000	-0.488002	0.465282	0.144652
mpg	-0.010774	-0.057093	0.125376	-0.488002	1.000000	-0.419417	-0.369919
engineSize	0.011365	0.014623	-0.006459	0.465282	-0.419417	1.000000	0.624899
price	0.012129	0.501296	-0.478705	0.144652	-0.369919	0.624899	1.000000

Let us compute the Variance Inflation Factor (VIF) for the four predictors.

```
X = train[['mpg','year','mileage','engineSize']]
```

```
X.columns[1:]
```

```
Index(['year', 'mileage', 'engineSize'], dtype='object')
```

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
X = add_constant(X)
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns

for i in range(len(X.columns)):
    vif_data.loc[i,'VIF'] = variance_inflation_factor(X.values, i)

print(vif_data)
```

```
      feature           VIF
0       const  1.201579e+06
1         mpg  1.243040e+00
2        year  2.452891e+00
3     mileage  2.490210e+00
4  engineSize  1.219170e+00
```

As all the values of VIF are close to one, we do not have the problem of multicollinearity in the model. Note that the VIF of `year` and `mileage` is relatively high as they are the most correlated.

Q1: Why is the VIF of the constant so high?

Q2: Why do we need to include the constant while finding the VIF?

5.4.3 Manual computation of VIF

```
#Manually computing the VIF for year
ols_object = smf.ols(formula = 'year~mpg+engineSize+mileage', data = train)
model_log = ols_object.fit()
model_log.summary()
```

Table 5.7: OLS Regression Results

Dep. Variable:	year	R-squared:	0.592
Model:	OLS	Adj. R-squared:	0.592
Method:	Least Squares	F-statistic:	2400.
Date:	Mon, 30 Jan 2023	Prob (F-statistic):	0.00
Time:	02:49:19	Log-Likelihood:	-10066.
No. Observations:	4960	AIC:	2.014e+04
Df Residuals:	4956	BIC:	2.017e+04
Df Model:	3		
Covariance Type:	nonrobust		

```
#VIF for year
1/(1-0.592)
```

```
2.4509803921568625
```

Note that year and mileage have a high linear correlation. Removing one of them should decrease the standard error of the coefficient of the other, without significanty decrease R-squared.

```
ols_object = smf.ols(formula = 'price~mpg+engineSize+mileage+year', data = train)
model_log = ols_object.fit()
model_log.summary()
```

Table 5.8: OLS Regression Results

Dep. Variable:	price	R-squared:	0.660
Model:	OLS	Adj. R-squared:	0.660
Method:	Least Squares	F-statistic:	2410.
Date:	Tue, 07 Feb 2023	Prob (F-statistic):	0.00
Time:	21:39:45	Log-Likelihood:	-52497.
No. Observations:	4960	AIC:	1.050e+05
Df Residuals:	4955	BIC:	1.050e+05
Df Model:	4		
Covariance Type:	nonrobust		

Removing mileage from the above regression.

```
ols_object = smf.ols(formula = 'price~mpg+engineSize+year', data = train)
model_log = ols_object.fit()
model_log.summary()
```

Table 5.9: OLS Regression Results

Dep. Variable:	price	R-squared:	0.641
Model:	OLS	Adj. R-squared:	0.641
Method:	Least Squares	F-statistic:	2951.
Date:	Tue, 07 Feb 2023	Prob (F-statistic):	0.00
Time:	21:40:00	Log-Likelihood:	-52635.
No. Observations:	4960	AIC:	1.053e+05
Df Residuals:	4956	BIC:	1.053e+05
Df Model:	3		
Covariance Type:	nonrobust		

Note that the standard error of the coefficient of *year* has reduced from 73 to 48, without any large reduction in R-squared.

5.4.4 When can we overlook multicollinearity?

- The severity of the problems increases with the degree of the multicollinearity. Therefore, if there is only moderate multicollinearity *(5 < VIF < 10)*, we may overlook it.
- Multicollinearity affects only the standard errors of the coefficients of collinear predictors. Therefore, if multicollinearity is not present for the predictors that we are particularly interested in, we may not need to resolve it.

- Multicollinearity affects the standard error of the coefficients and thereby their p-values, but in general, it does not influence the prediction accuracy, except in the case that the coefficients are so unstable that the predictions are outside of the domain space of the response. If our sole aim is prediction, and we don't wish to infer the statistical significance of predictors, then we may avoid addressing multicollinearity. "*The fact that some or all predictor variables are correlated among themselves does not, in general, inhibit our ability to obtain a good fit nor does it tend to affect inferences about mean responses or predictions of new observations, provided these inferences are made within the region of observations" - Neter, John, Michael H. Kutner, Christopher J. Nachtsheim, and William Wasserman. "Applied linear statistical models." (1996): 318.*

6 Autocorrelation

Read section 3.3.3 (2) of the book before using these notes.

Note that in this course, lecture notes are not sufficient, you must read the book for better understanding. Lecture notes are just implementing the concepts of the book on a dataset, but not explaining the concepts elaborately.

Below is an example showing violation of the autocorrelation assumption *(refer to the book to understand autocorrelation)* in linear regression. Subsequently, it is shown that addressing the assumption violation leads to a much better model fit.

Example: Using linear regression models to predict electricity demand in Toronto, CA.

We have hourly power demand and temperature (in Celsius) data from 2017 to 2020.

We are going to build a linear model to predict the hourly power demand for the next day (for example, when it is 1/1/2021, we predict hourly demand on 1/2/2021 using historical data and the weather forecasts).

When we are building a model, it is important to keep in mind what data we can use as features. For this model:

- We cannot use previous hourly data as features. (Although in a high frequency setting, it is possible)
- The temperature in our raw data can not be used directly, since it is the actual, not the forecasted temperature. We are going to use the previous day temperature as the forecast.

Source: Keep it simple, keep it linear: A linear regression model for time series

```
%pylab inline
import pandas as pd
import seaborn as sns
import statsmodels.api as sm
plt.rcParams['figure.figsize'] = [9, 5]
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
# A few helper functions
import numpy.ma as ma
from scipy.stats.stats import pearsonr, normaltest
from scipy.spatial.distance import correlation
def build_model(features):
  X=sm.add_constant(df[features])
  y=df['power']
  model = sm.OLS(y,X, missing='drop').fit()
  predictions = model.predict(X)
  display(model.summary())
  res=y-predictions
  return res


def plt_residual(res):
  plt.plot(range(len(res)), res)
  plt.ylabel('Residual')
  plt.xlabel("Hour")

def plt_residual_lag(res, nlag):
  x=res.values
  y=res.shift(nlag).values
  sns.kdeplot(x,y=y,color='blue',shade=True )
  plt.xlabel('res')
  plt.ylabel("res-lag-{}".format(nlag))
  rho,p=corrcoef(x,y)
  plt.title("n_lag={} hours, correlation={:f}".format(nlag, rho))

def plt_acf(res):
  plt.rcParams['figure.figsize'] = [18, 5]
  acorr = sm.tsa.acf(res.dropna(), nlags = len(res.dropna())-1)
  fig, (ax1, ax2) = plt.subplots(1, 2)
  ax1.plot(acorr)
  ax1.set_ylabel('corr')
  ax1.set_xlabel('n_lag')
  ax1.set_title('Auto Correlation')
  ax2.plot(acorr[:4*7*24])
  ax2.set_ylabel('corr')
  ax2.set_xlabel('n_lag')
  ax2.set_title('Auto Correlation (4-week zoomed in) ')
  plt.show()
```

```
  pd.set_option('display.max_columns', None)
  adf=pd.DataFrame(np.round(acorr[:30*24],2).reshape([30, 24] ))
  adf.index.name='day'
  display(adf)
  plt.rcParams['figure.figsize'] = [9, 5]

def corrcoef(x,y):
    a,b=ma.masked_invalid(x),ma.masked_invalid(y)
    msk = (~a.mask & ~b.mask)
    return pearsonr(x[msk],y[msk])[0], normaltest(res, nan_policy='omit')[1]
```

6.1 The data

```
df=pd.read_csv("./Datasets/Toronto_power_demand.csv", parse_dates=['Date'], index_col=0)
df['temperature']=df['temperature'].shift(24*1)
df.tail()
```

key	Date	Hour	power	temperature
20201231:19	2020-12-31	19	5948	4.9
20201231:20	2020-12-31	20	5741	4.5
20201231:21	2020-12-31	21	5527	3.7
20201231:22	2020-12-31	22	5301	2.9
20201231:23	2020-12-31	23	5094	2.1

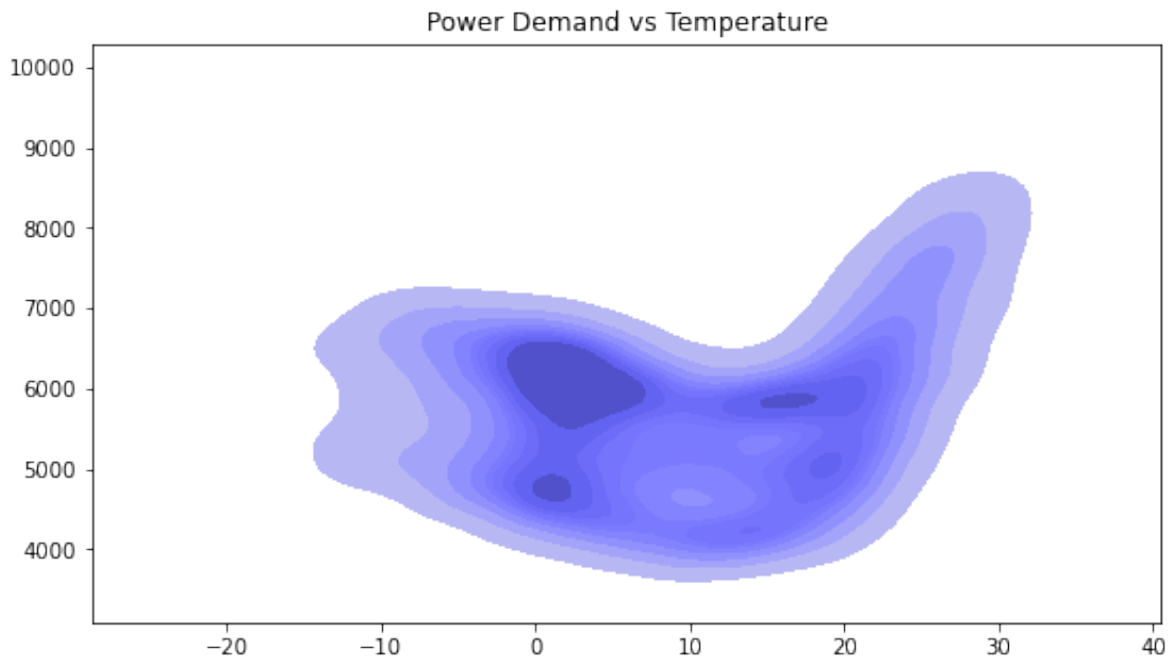
```
ndays=len(set(df['Date']))
print("There are {} rows, which is {}*24={}, for {} days. And The data is already in sorte
```

```
There are 35064 rows, which is 1461*24=35064, for 1461 days. And The data is already in sorte
```

```
print("It is natural to think that there is a relationship between power demand and temper
sns.kdeplot(df['temperature'].values, y=df['power'].values,color='blue',shade=True )
plt.title("Power Demand vs Temperature")
```

```
It is natural to think that there is a relationship between power demand and temperature.
```

```
Text(0.5, 1.0, 'Power Demand vs Temperature')
```




```
print("""
It is not a linear relationship. We create two features corresponding to hot and cold weat
it possible to develop a linear model.
""")
is_hot=(df['temperature']>15).astype(int)
print("{:f}% of data points are hot".format(is_hot.mean()*100))
df['temp_hot']=df['temperature']*is_hot
df['temp_cold']=df['temperature']*(1-is_hot)
df.tail()
```

```
It is not a linear relationship. We create two features corresponding to hot and cold weathe

34.813484% of data points are hot
```

key	Date	Hour	power	temperature	temp_hot	temp_cold
20201231:19	2020-12-31	19	5948	4.9	0.0	4.9
20201231:20	2020-12-31	20	5741	4.5	0.0	4.5
20201231:21	2020-12-31	21	5527	3.7	0.0	3.7
20201231:22	2020-12-31	22	5301	2.9	0.0	2.9
20201231:23	2020-12-31	23	5094	2.1	0.0	2.1

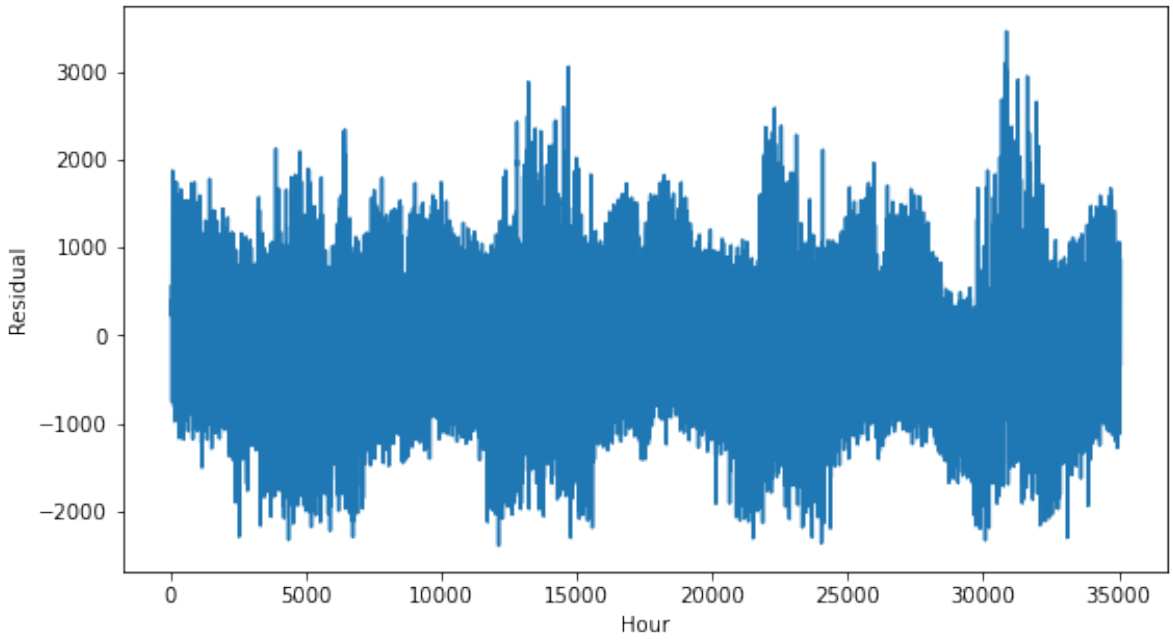
6.2 Predictor: temperature

```
res=build_model(['temp_hot', 'temp_cold'])
```

Table 6.3: OLS Regression Results

Dep. Variable:	power	R-squared:	0.195
Model:	OLS	Adj. R-squared:	0.195
Method:	Least Squares	F-statistic:	4251.
Date:	Sun, 05 Feb 2023	Prob (F-statistic):	0.00
Time:	23:15:53	Log-Likelihood:	-2.8766e+05
No. Observations:	35040	AIC:	5.753e+05
Df Residuals:	35037	BIC:	5.753e+05
Df Model:	2		
Covariance Type:	nonrobust		

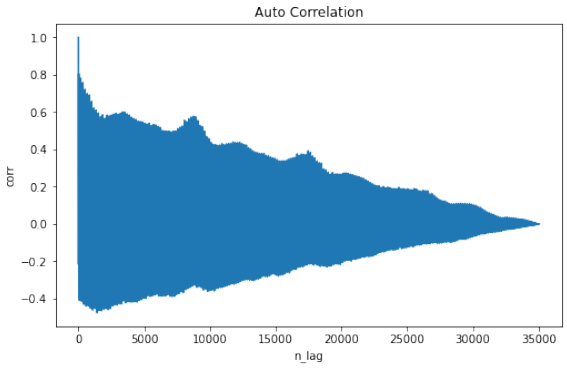
```
plt_residual(res)
```

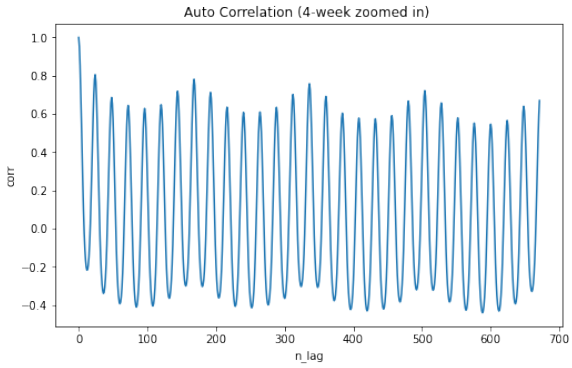



```
print("acf shows that there is a strong correlation for 24 lags, which is one day.")
plt_acf(res)
```

```
acf shows that there is a strong correlation for 24 lags, which is one day.
```

```
C:\Users\akl0407\Anaconda3\lib\site-packages\statsmodels\tsa\stattools.py:667: FutureWarning
  warnings.warn(
```

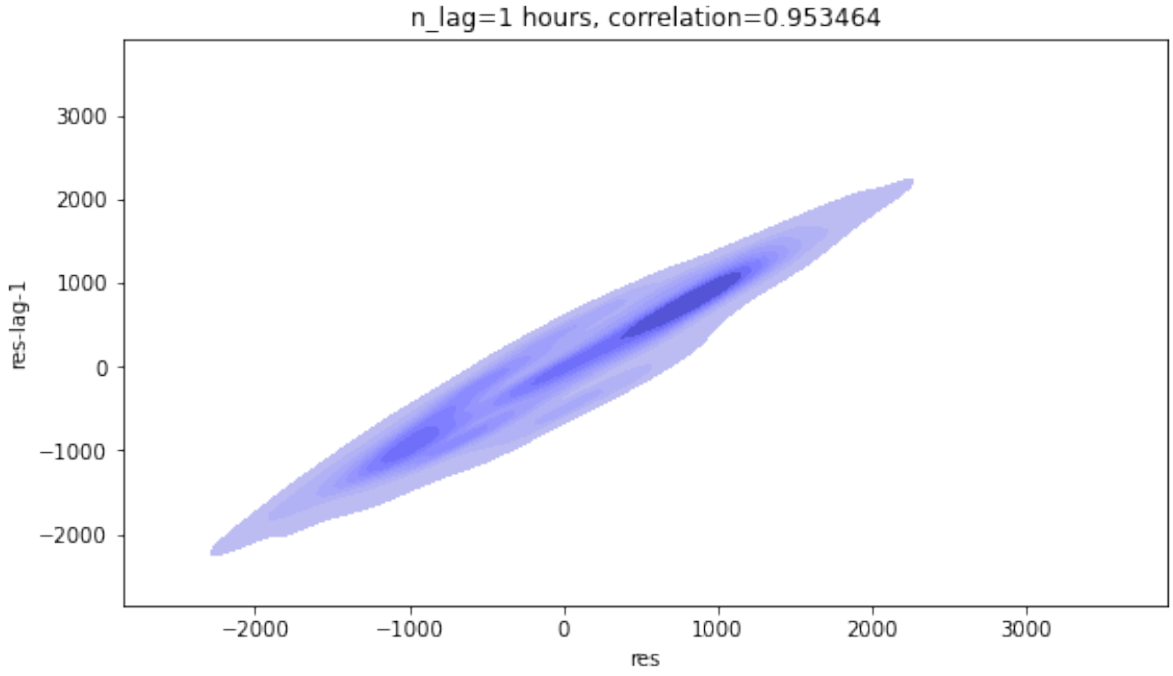


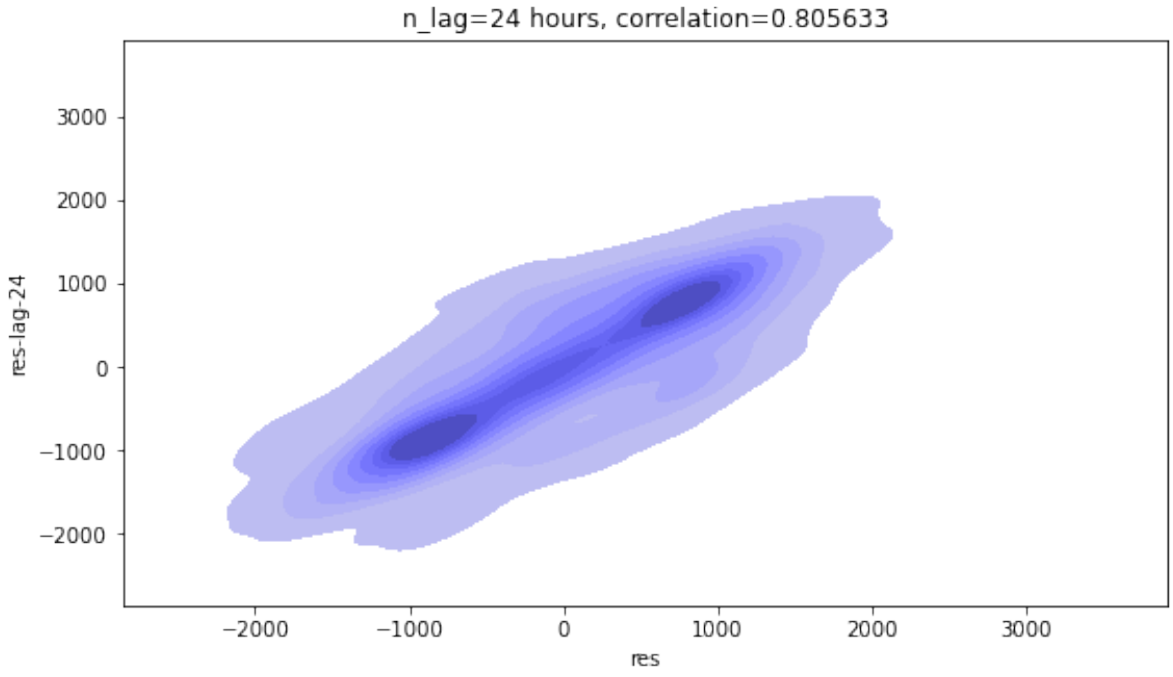


day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1.00	0.95	0.85	0.72	0.56	0.40	0.24	0.09	-0.02	-0.11	-0.16	-0.20	-0.22	-0.21	-0.19
1	0.81	0.77	0.68	0.55	0.40	0.25	0.09	-0.04	-0.15	-0.23	-0.29	-0.32	-0.34	-0.33	-0.31
2	0.69	0.65	0.57	0.45	0.31	0.16	0.01	-0.12	-0.22	-0.30	-0.35	-0.38	-0.39	-0.38	-0.36
3	0.64	0.61	0.53	0.42	0.28	0.13	-0.01	-0.14	-0.24	-0.32	-0.37	-0.40	-0.41	-0.40	-0.37
4	0.63	0.60	0.52	0.41	0.27	0.12	-0.02	-0.14	-0.24	-0.32	-0.37	-0.40	-0.40	-0.39	-0.36
5	0.65	0.62	0.54	0.43	0.30	0.15	0.01	-0.11	-0.21	-0.28	-0.33	-0.36	-0.36	-0.35	-0.32
6	0.72	0.69	0.61	0.50	0.36	0.21	0.07	-0.05	-0.15	-0.22	-0.27	-0.29	-0.30	-0.29	-0.26
7	0.78	0.75	0.67	0.54	0.40	0.25	0.10	-0.03	-0.13	-0.21	-0.26	-0.29	-0.30	-0.30	-0.27
8	0.71	0.68	0.60	0.48	0.34	0.19	0.04	-0.09	-0.19	-0.27	-0.32	-0.35	-0.36	-0.36	-0.33
9	0.64	0.61	0.53	0.41	0.27	0.13	-0.02	-0.14	-0.24	-0.32	-0.37	-0.40	-0.41	-0.40	-0.37
10	0.61	0.58	0.50	0.39	0.25	0.11	-0.03	-0.16	-0.26	-0.33	-0.38	-0.40	-0.41	-0.40	-0.38
11	0.61	0.58	0.51	0.39	0.26	0.12	-0.02	-0.14	-0.24	-0.32	-0.36	-0.39	-0.40	-0.39	-0.36
12	0.63	0.61	0.53	0.42	0.28	0.14	0.00	-0.12	-0.22	-0.29	-0.33	-0.36	-0.36	-0.35	-0.32
13	0.70	0.67	0.60	0.48	0.35	0.20	0.06	-0.06	-0.16	-0.23	-0.27	-0.30	-0.30	-0.29	-0.26
14	0.76	0.73	0.64	0.52	0.38	0.23	0.09	-0.04	-0.14	-0.22	-0.27	-0.30	-0.31	-0.30	-0.27
15	0.69	0.66	0.58	0.46	0.32	0.17	0.03	-0.10	-0.20	-0.28	-0.33	-0.36	-0.38	-0.37	-0.35
16	0.60	0.57	0.50	0.38	0.25	0.10	-0.04	-0.16	-0.26	-0.34	-0.38	-0.41	-0.42	-0.41	-0.39
17	0.58	0.55	0.47	0.36	0.23	0.09	-0.05	-0.17	-0.27	-0.34	-0.39	-0.42	-0.43	-0.42	-0.39
18	0.57	0.55	0.47	0.36	0.23	0.09	-0.05	-0.17	-0.27	-0.34	-0.39	-0.41	-0.42	-0.41	-0.38
19	0.59	0.57	0.49	0.38	0.25	0.11	-0.03	-0.14	-0.24	-0.31	-0.35	-0.38	-0.38	-0.37	-0.34
20	0.67	0.64	0.56	0.45	0.32	0.18	0.04	-0.08	-0.17	-0.24	-0.29	-0.31	-0.32	-0.31	-0.28
21	0.72	0.69	0.61	0.49	0.36	0.21	0.07	-0.06	-0.16	-0.23	-0.28	-0.31	-0.32	-0.32	-0.29
22	0.66	0.63	0.55	0.43	0.29	0.15	0.01	-0.12	-0.22	-0.29	-0.34	-0.37	-0.38	-0.38	-0.35
23	0.58	0.55	0.47	0.36	0.23	0.09	-0.05	-0.17	-0.27	-0.34	-0.39	-0.42	-0.43	-0.42	-0.39
24	0.55	0.52	0.45	0.34	0.21	0.07	-0.07	-0.19	-0.29	-0.36	-0.40	-0.43	-0.44	-0.43	-0.40
25	0.55	0.52	0.45	0.34	0.21	0.07	-0.07	-0.19	-0.28	-0.35	-0.40	-0.42	-0.43	-0.42	-0.39
26	0.57	0.54	0.47	0.36	0.23	0.09	-0.04	-0.16	-0.25	-0.32	-0.36	-0.39	-0.39	-0.38	-0.35
27	0.64	0.61	0.54	0.43	0.30	0.16	0.03	-0.09	-0.19	-0.25	-0.30	-0.32	-0.33	-0.32	-0.29
28	0.70	0.67	0.59	0.47	0.34	0.19	0.05	-0.07	-0.17	-0.24	-0.29	-0.32	-0.33	-0.33	-0.30
29	0.63	0.60	0.53	0.41	0.28	0.13	-0.01	-0.13	-0.23	-0.30	-0.35	-0.38	-0.39	-0.39	-0.37

```
print("Although 1 hour lag correlation is more strong, but we cannot use it, as we intend
the power consumption for the next day.")
plt_residual_lag(res,1)
plt.show()
plt_residual_lag(res,24)
```

```
Although 1 hour lag correlation is more strong, but we cannot use it, as we intend to predict
```


n_lag=1 hours, correlation=0.953464
res-lag-1
res
3000
2000
1000
0
−1000
−2000
−2000
−1000
0
1000
2000
3000


n_lag=24 hours, correlation=0.805633
res-lag-24
res
3000
2000
1000
0
−1000
−2000
−2000
−1000
0
1000
2000
3000

6.3 Predictors: Temperature + one day lag of power.

```
df['power_lag_1_day']=df['power'].shift(24)
df.tail()
```

	key	Date	Hour	power	temperature	temp_hot	temp_cold	power_lag_1_day
35059	20201231:19	2020-12-31	19	5948	4.9	0.0	4.9	6163.0
35060	20201231:20	2020-12-31	20	5741	4.5	0.0	4.5	5983.0
35061	20201231:21	2020-12-31	21	5527	3.7	0.0	3.7	5727.0
35062	20201231:22	2020-12-31	22	5301	2.9	0.0	2.9	5428.0
35063	20201231:23	2020-12-31	23	5094	2.1	0.0	2.1	5104.0

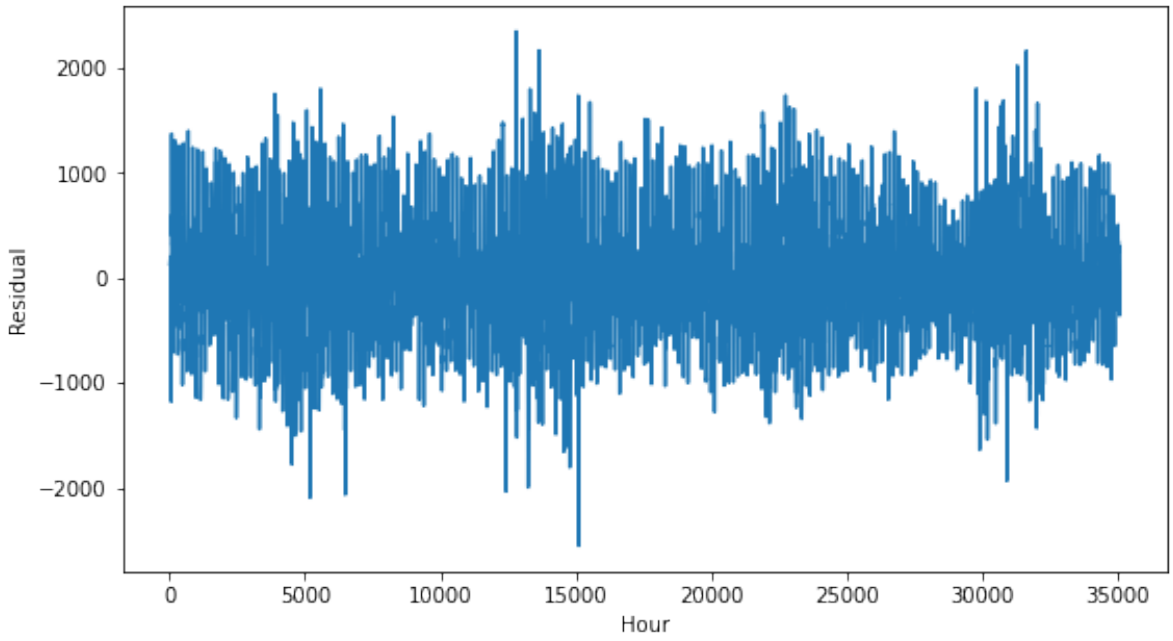
```
res=build_model(['temp_hot', 'temp_cold', 'power_lag_1_day' ])
```

```
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/tsatools.py:142: FutureWarning: In a f
  x = pd.concat(x[::order], 1)
```

Table 6.6: OLS Regression Results

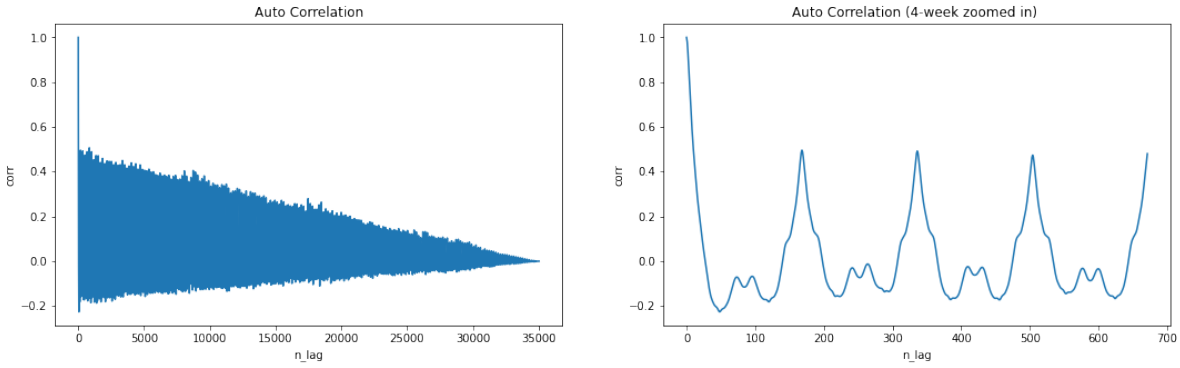
Dep. Variable:	power	R-squared:	0.794
Model:	OLS	Adj. R-squared:	0.794
Method:	Least Squares	F-statistic:	4.513e+04
Date:	Sun, 22 Jan 2023	Prob (F-statistic):	0.00
Time:	19:21:14	Log-Likelihood:	-2.6375e+05
No. Observations:	35040	AIC:	5.275e+05
Df Residuals:	35036	BIC:	5.275e+05
Df Model:	3		
Covariance Type:	nonrobust		

```
plt_residual(res)
```




```
plt_acf(res)
```

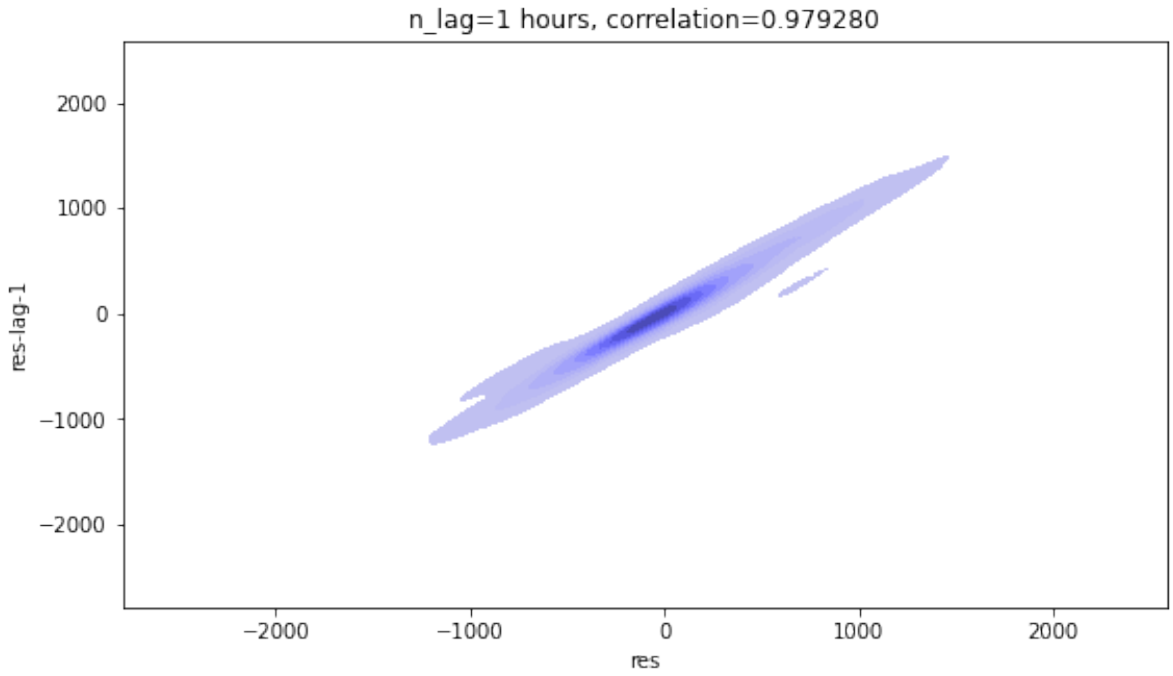
```
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/stattools.py:667: FutureWarning: fft=T
  warnings.warn(
```



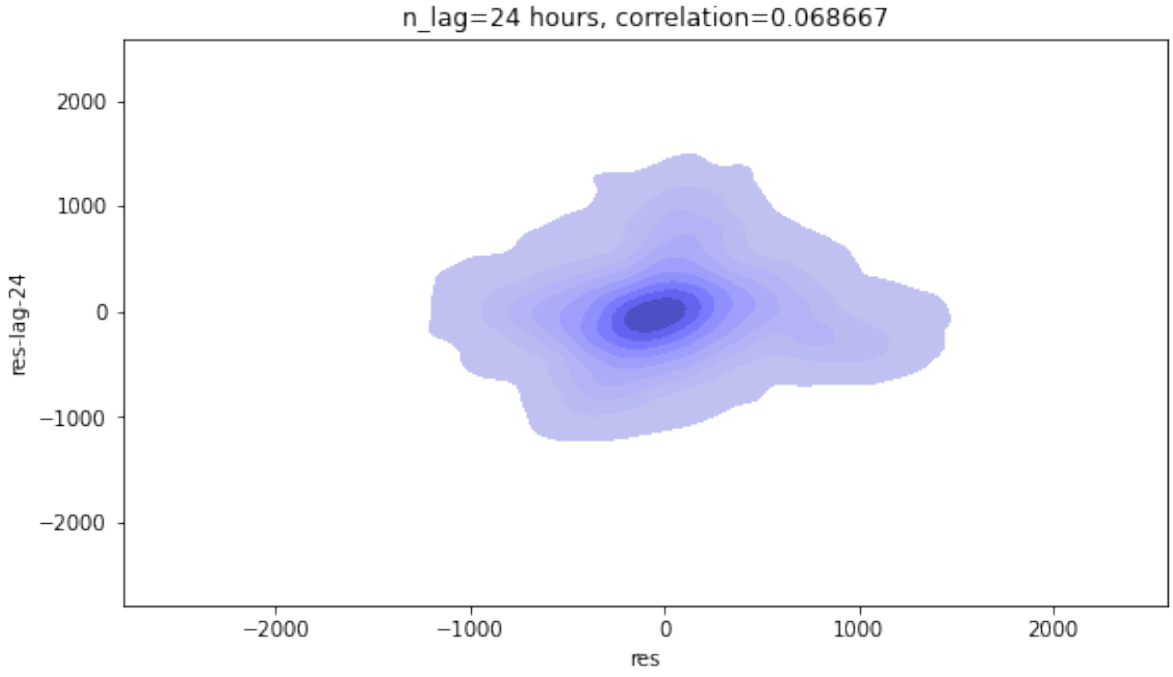
day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1.00	0.98	0.93	0.87	0.81	0.75	0.70	0.64	0.59	0.54	0.50	0.46	0.42	0.39	0.

day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.07	0.05	0.03	0.00	-0.02	-0.04	-0.06	-0.08	-0.10	-0.12	-0.14	-0.15	-0.16	-0.17	-0
2	-0.23	-0.23	-0.22	-0.22	-0.21	-0.21	-0.21	-0.21	-0.21	-0.20	-0.20	-0.19	-0.18	-0.18	-0
3	-0.07	-0.07	-0.07	-0.07	-0.08	-0.09	-0.09	-0.10	-0.11	-0.11	-0.11	-0.12	-0.12	-0.12	-0
4	-0.07	-0.07	-0.07	-0.08	-0.09	-0.10	-0.11	-0.12	-0.13	-0.14	-0.14	-0.15	-0.16	-0.16	-0
5	-0.18	-0.18	-0.17	-0.17	-0.17	-0.16	-0.16	-0.16	-0.16	-0.15	-0.14	-0.14	-0.13	-0.12	-0
6	0.07	0.08	0.09	0.09	0.10	0.10	0.11	0.12	0.13	0.14	0.16	0.18	0.19	0.21	0.
7	0.50	0.49	0.46	0.43	0.40	0.37	0.34	0.31	0.28	0.26	0.24	0.22	0.21	0.19	0.
8	0.10	0.09	0.07	0.06	0.04	0.02	-0.00	-0.02	-0.04	-0.05	-0.07	-0.08	-0.09	-0.10	-0
9	-0.16	-0.16	-0.16	-0.15	-0.16	-0.16	-0.16	-0.16	-0.16	-0.16	-0.15	-0.15	-0.14	-0.14	-0
10	-0.03	-0.03	-0.03	-0.03	-0.04	-0.04	-0.05	-0.06	-0.06	-0.07	-0.07	-0.07	-0.07	-0.07	-0
11	-0.01	-0.01	-0.02	-0.03	-0.03	-0.04	-0.05	-0.06	-0.07	-0.08	-0.09	-0.10	-0.11	-0.11	-0
12	-0.14	-0.14	-0.13	-0.13	-0.13	-0.13	-0.14	-0.14	-0.13	-0.13	-0.13	-0.12	-0.11	-0.10	-0
13	0.08	0.09	0.10	0.10	0.11	0.11	0.12	0.13	0.14	0.15	0.17	0.18	0.20	0.22	0.
14	0.49	0.48	0.46	0.43	0.40	0.37	0.34	0.31	0.28	0.26	0.24	0.23	0.21	0.20	0.
15	0.10	0.09	0.07	0.05	0.03	0.01	-0.01	-0.03	-0.05	-0.07	-0.08	-0.10	-0.11	-0.12	-0
16	-0.17	-0.17	-0.17	-0.17	-0.16	-0.16	-0.17	-0.17	-0.16	-0.16	-0.16	-0.16	-0.15	-0.14	-0
17	-0.03	-0.02	-0.02	-0.03	-0.03	-0.04	-0.04	-0.05	-0.05	-0.06	-0.06	-0.06	-0.06	-0.06	-0
18	-0.03	-0.04	-0.04	-0.05	-0.06	-0.08	-0.09	-0.10	-0.11	-0.12	-0.13	-0.14	-0.15	-0.15	-0
19	-0.17	-0.17	-0.16	-0.16	-0.15	-0.15	-0.15	-0.15	-0.14	-0.13	-0.13	-0.12	-0.11	-0.10	-0
20	0.10	0.10	0.11	0.11	0.12	0.12	0.12	0.13	0.14	0.15	0.17	0.18	0.20	0.21	0.
21	0.47	0.46	0.44	0.41	0.38	0.35	0.32	0.29	0.26	0.24	0.22	0.21	0.19	0.18	0.
22	0.10	0.09	0.07	0.05	0.03	0.01	-0.00	-0.02	-0.04	-0.05	-0.07	-0.08	-0.09	-0.10	-0
23	-0.14	-0.14	-0.14	-0.14	-0.14	-0.14	-0.14	-0.14	-0.14	-0.14	-0.13	-0.13	-0.13	-0.12	-0
24	-0.03	-0.03	-0.03	-0.04	-0.05	-0.05	-0.06	-0.07	-0.08	-0.08	-0.09	-0.09	-0.09	-0.09	-0
25	-0.03	-0.04	-0.04	-0.05	-0.06	-0.07	-0.08	-0.09	-0.10	-0.11	-0.12	-0.13	-0.14	-0.14	-0
26	-0.17	-0.17	-0.16	-0.16	-0.16	-0.15	-0.15	-0.15	-0.15	-0.14	-0.13	-0.13	-0.12	-0.11	-0
27	0.08	0.09	0.10	0.11	0.11	0.12	0.12	0.13	0.14	0.16	0.18	0.19	0.21	0.22	0.
28	0.49	0.48	0.45	0.42	0.39	0.36	0.33	0.30	0.27	0.25	0.23	0.22	0.20	0.19	0.
29	0.09	0.08	0.06	0.04	0.02	-0.00	-0.02	-0.04	-0.06	-0.08	-0.09	-0.10	-0.12	-0.13	-0

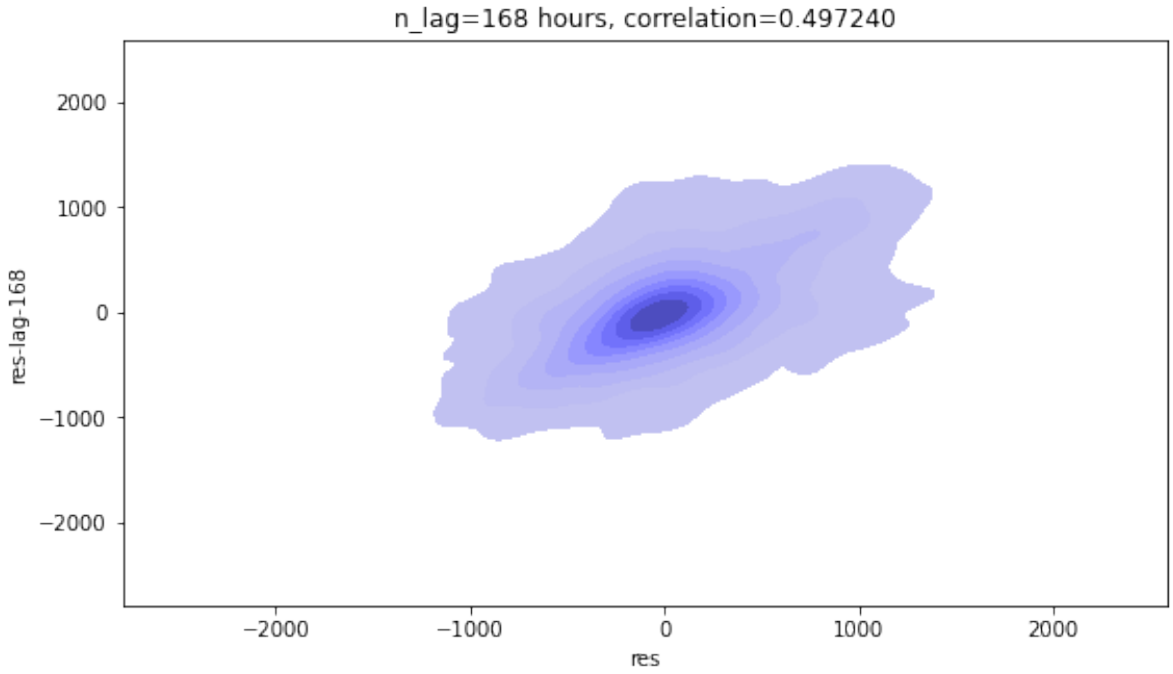
```
plt_residual_lag(res, 1)
```




```
plt_residual_lag(res, 24)
```



```
plt_residual_lag(res, 24*7)
```



6.4 Predictors: Temperature + 1 day lag of power + 1 week lag of power

```
df['power_lag_1_week']=df['power'].shift(24*7)
df.tail()
```

	key	Date	Hour	power	temperature	temp_hot	temp_cold	power_lag_1_day
35059	20201231:19	2020-12-31	19	5948	4.9	0.0	4.9	6163.0
35060	20201231:20	2020-12-31	20	5741	4.5	0.0	4.5	5983.0
35061	20201231:21	2020-12-31	21	5527	3.7	0.0	3.7	5727.0
35062	20201231:22	2020-12-31	22	5301	2.9	0.0	2.9	5428.0
35063	20201231:23	2020-12-31	23	5094	2.1	0.0	2.1	5104.0

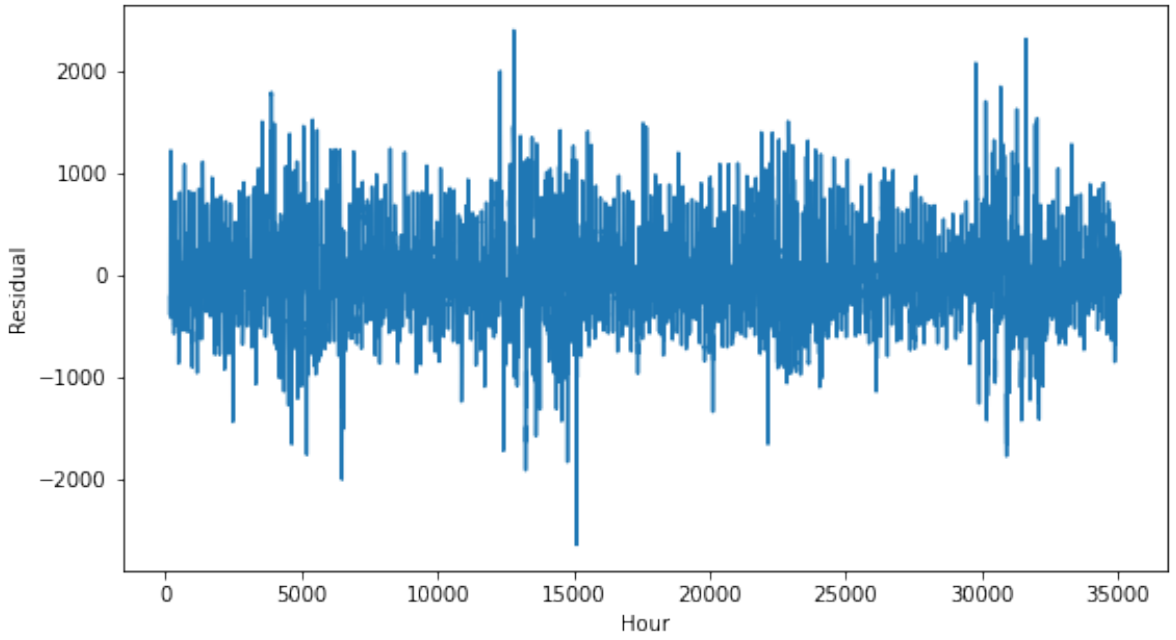
```
res=build_model(['temp_hot', 'temp_cold', 'power_lag_1_day', 'power_lag_1_week' ])
```

```
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/tsatools.py:142: FutureWarning: In a f
  x = pd.concat(x[::order], 1)
```

Table 6.9: OLS Regression Results

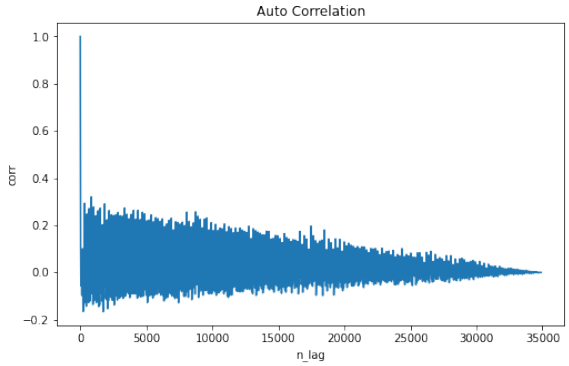
Dep. Variable:	power	R-squared:	0.840
Model:	OLS	Adj. R-squared:	0.840
Method:	Least Squares	F-statistic:	4.585e+04
Date:	Sun, 22 Jan 2023	Prob (F-statistic):	0.00
Time:	19:22:49	Log-Likelihood:	-2.5830e+05
No. Observations:	34896	AIC:	5.166e+05
Df Residuals:	34891	BIC:	5.167e+05
Df Model:	4		
Covariance Type:	nonrobust		

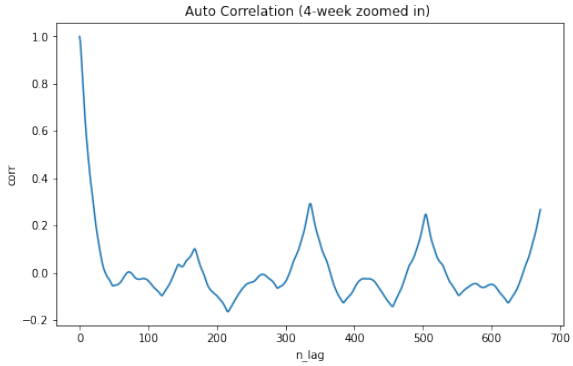
```
plt_residual(res)
```




```
plt_acf(res)
```

```
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/stattools.py:667: FutureWarning: fft=T
  warnings.warn(
```

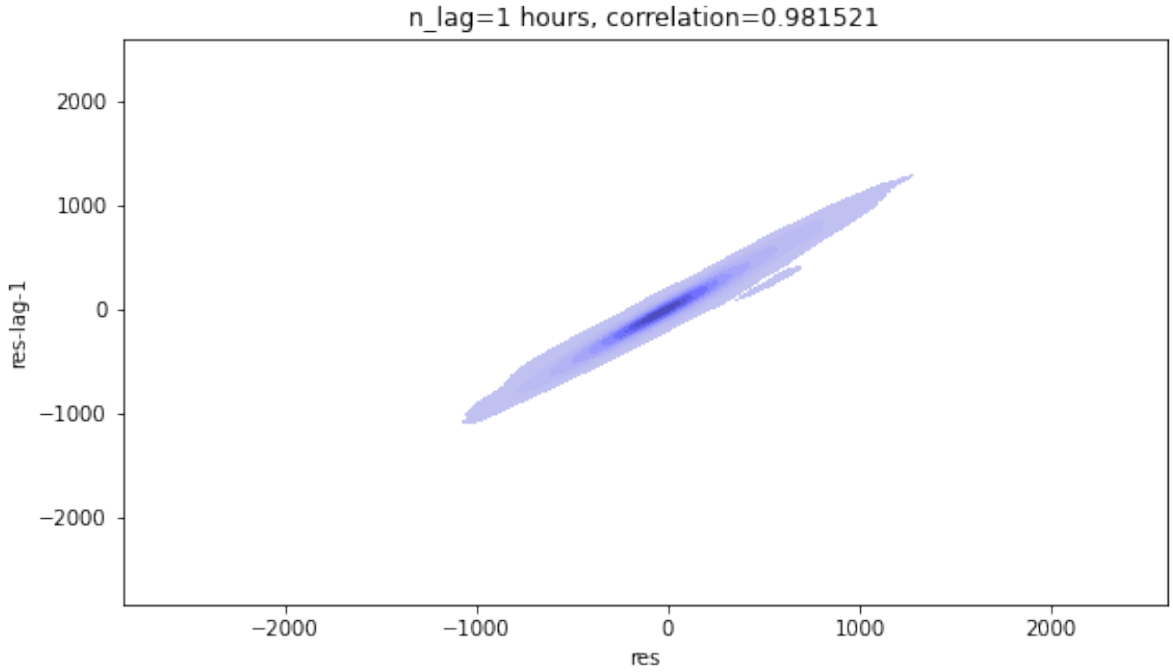




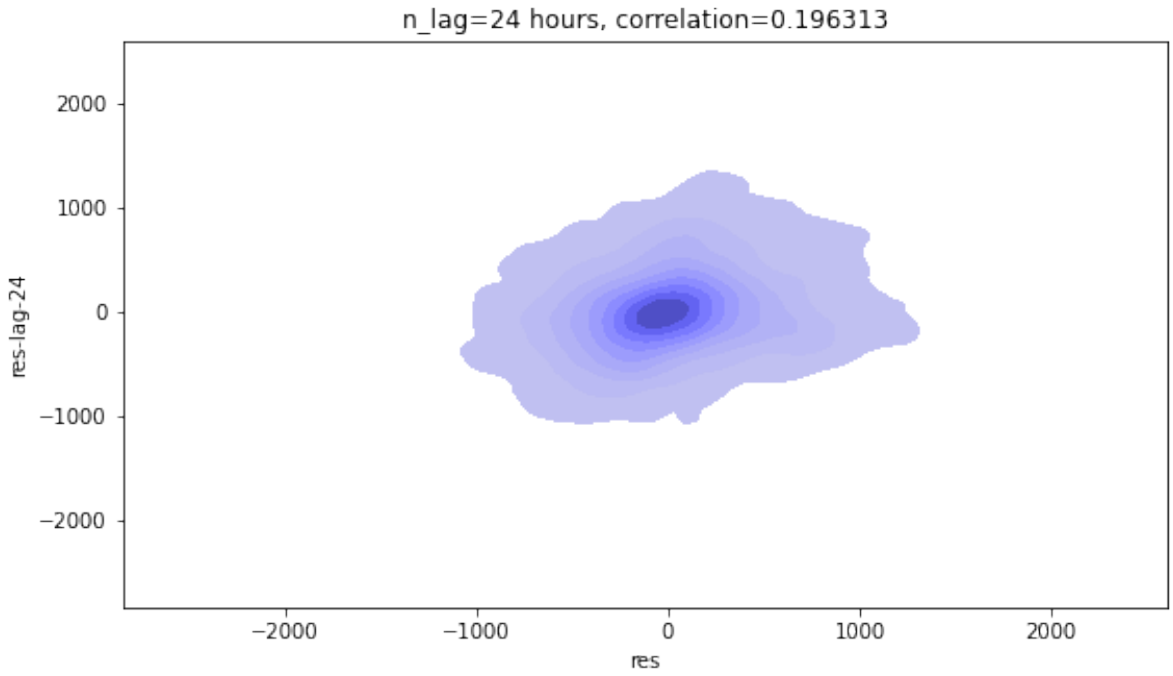
day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1.00	0.98	0.94	0.89	0.84	0.79	0.74	0.70	0.65	0.61	0.58	0.54	0.51	0.48	0.4
1	0.20	0.18	0.16	0.14	0.12	0.10	0.09	0.07	0.06	0.04	0.03	0.02	0.01	0.00	-0
2	-0.05	-0.06	-0.06	-0.05	-0.05	-0.05	-0.05	-0.05	-0.05	-0.05	-0.05	-0.04	-0.04	-0.04	-0
3	0.00	0.00	0.00	-0.00	-0.00	-0.01	-0.01	-0.01	-0.02	-0.02	-0.02	-0.03	-0.03	-0.03	-0
4	-0.03	-0.03	-0.03	-0.03	-0.03	-0.04	-0.04	-0.04	-0.05	-0.05	-0.05	-0.06	-0.06	-0.06	-0
5	-0.10	-0.09	-0.09	-0.08	-0.08	-0.07	-0.07	-0.07	-0.06	-0.06	-0.05	-0.05	-0.04	-0.04	-0
6	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.04	0.05	0.05	0.05	0.0
7	0.10	0.10	0.09	0.08	0.06	0.05	0.05	0.04	0.03	0.02	0.01	0.01	0.00	-0.01	-0
8	-0.08	-0.08	-0.08	-0.09	-0.09	-0.09	-0.09	-0.10	-0.10	-0.10	-0.10	-0.11	-0.11	-0.11	-0
9	-0.17	-0.16	-0.16	-0.15	-0.15	-0.14	-0.14	-0.13	-0.13	-0.12	-0.12	-0.11	-0.11	-0.10	-0
10	-0.06	-0.05	-0.05	-0.05	-0.05	-0.05	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0
11	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.02	-0.02	-0.02	-0.03	-0.03	-0
12	-0.07	-0.07	-0.06	-0.06	-0.06	-0.06	-0.06	-0.05	-0.05	-0.05	-0.05	-0.04	-0.04	-0.03	-0
13	0.05	0.06	0.06	0.07	0.08	0.08	0.09	0.10	0.11	0.12	0.13	0.14	0.15	0.16	0.1
14	0.29	0.29	0.27	0.26	0.24	0.23	0.21	0.20	0.19	0.18	0.17	0.16	0.15	0.14	0.1
15	0.05	0.04	0.03	0.02	0.01	-0.00	-0.01	-0.02	-0.03	-0.04	-0.05	-0.06	-0.06	-0.07	-0
16	-0.13	-0.13	-0.12	-0.12	-0.11	-0.11	-0.11	-0.10	-0.10	-0.10	-0.10	-0.09	-0.09	-0.08	-0
17	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0
18	-0.05	-0.05	-0.06	-0.06	-0.06	-0.07	-0.07	-0.08	-0.08	-0.09	-0.09	-0.10	-0.10	-0.10	-0
19	-0.14	-0.14	-0.13	-0.13	-0.12	-0.11	-0.11	-0.10	-0.10	-0.09	-0.08	-0.08	-0.07	-0.06	-0
20	0.03	0.03	0.04	0.04	0.05	0.05	0.06	0.07	0.07	0.08	0.09	0.10	0.11	0.12	0.1
21	0.25	0.24	0.23	0.22	0.20	0.19	0.17	0.16	0.14	0.13	0.12	0.11	0.11	0.10	0.0
22	0.04	0.03	0.02	0.02	0.01	0.00	-0.01	-0.01	-0.02	-0.03	-0.03	-0.04	-0.04	-0.05	-0
23	-0.10	-0.10	-0.09	-0.09	-0.09	-0.08	-0.08	-0.08	-0.07	-0.07	-0.07	-0.07	-0.07	-0.06	-0
24	-0.05	-0.05	-0.05	-0.05	-0.05	-0.05	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0.06	-0
25	-0.05	-0.05	-0.05	-0.05	-0.05	-0.06	-0.06	-0.06	-0.07	-0.07	-0.08	-0.08	-0.09	-0.09	-0
26	-0.13	-0.13	-0.12	-0.12	-0.11	-0.11	-0.10	-0.10	-0.09	-0.09	-0.08	-0.08	-0.07	-0.07	-0
27	0.02	0.03	0.03	0.04	0.05	0.06	0.06	0.07	0.08	0.10	0.11	0.12	0.13	0.14	0.1

day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
28	0.27	0.27	0.25	0.24	0.22	0.21	0.19	0.18	0.17	0.15	0.14	0.14	0.13	0.12	0.
29	0.04	0.03	0.02	0.02	0.01	-0.00	-0.01	-0.02	-0.03	-0.04	-0.04	-0.05	-0.06	-0.07	-0

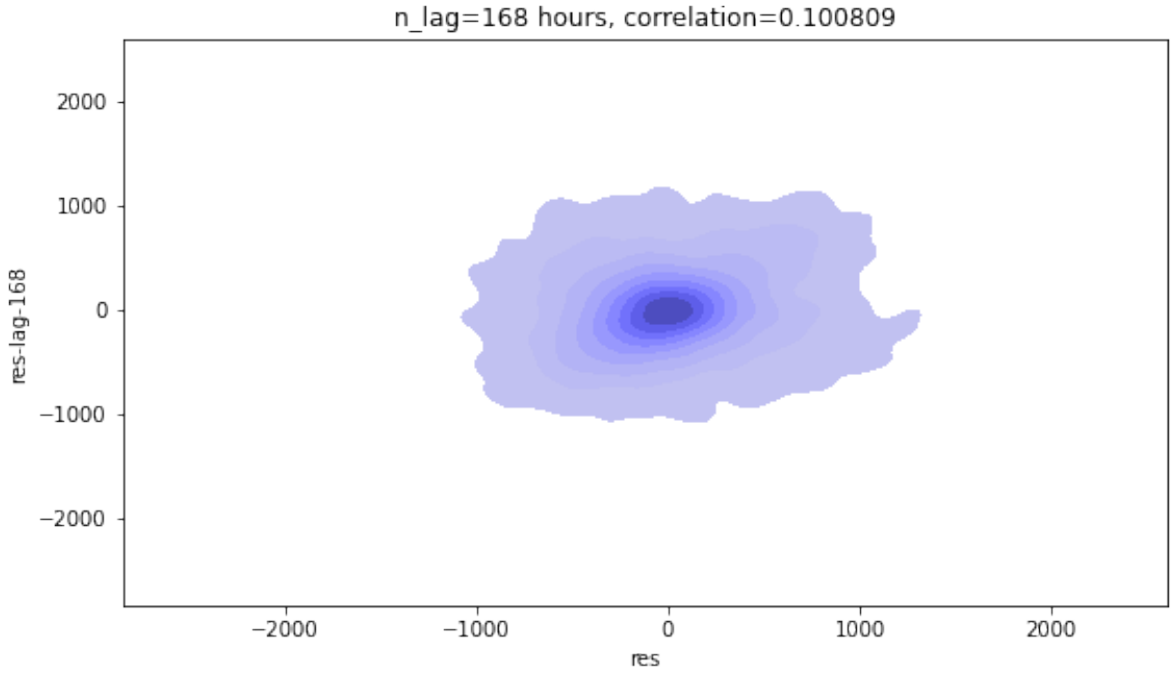
```
plt_residual_lag(res, 1)
```



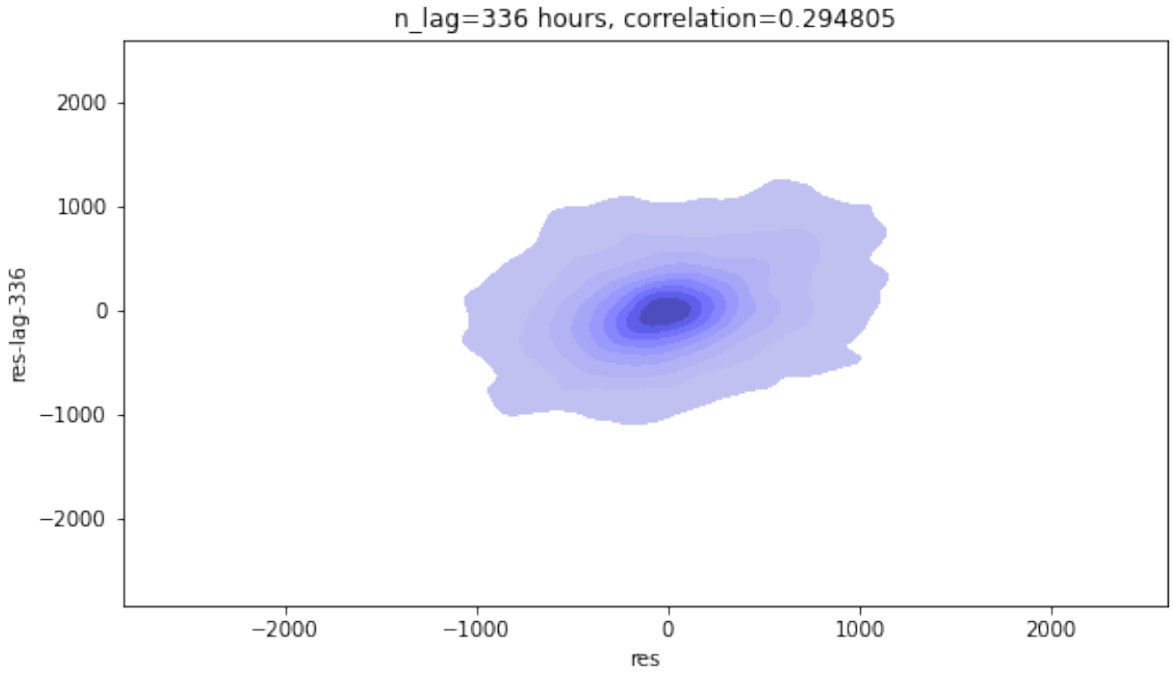

```
plt_residual_lag(res, 24)
```




```
plt_residual_lag(res, 24*7)
```



```
plt_residual_lag(res, 24*7*2)
```



6.5 Predictors: Temperature + 1 day lag of power + 1 week lag of power + 2 weeks lag of power

Although the data shows there is a significant (but not strong) correlation, we need to be cautious to use this feature because there are no simple reasons for this relationship.

For 1-day-lag feature, the correlation is easily understood.

For 1-week-lag feature, we could argue that the behaviour is different between weekday and weekend.

But for 2-week-lag feature, it is hard to understand especially when we have included 1-day-lag and 1-week-lag features. The relation is spurious.

```
df['power_lag_2_week']=df['power'].shift(24*7*2)
df.tail()
```

	key	Date	Hour	power	temperature	temp_hot	temp_cold	power_lag_1_day
35059	20201231:19	2020-12-31	19	5948	4.9	0.0	4.9	6163.0
35060	20201231:20	2020-12-31	20	5741	4.5	0.0	4.5	5983.0
35061	20201231:21	2020-12-31	21	5527	3.7	0.0	3.7	5727.0
35062	20201231:22	2020-12-31	22	5301	2.9	0.0	2.9	5428.0
35063	20201231:23	2020-12-31	23	5094	2.1	0.0	2.1	5104.0

```
res=build_model(['temp_hot', 'temp_cold', 'power_lag_1_day','power_lag_1_week', 'power_lag
```

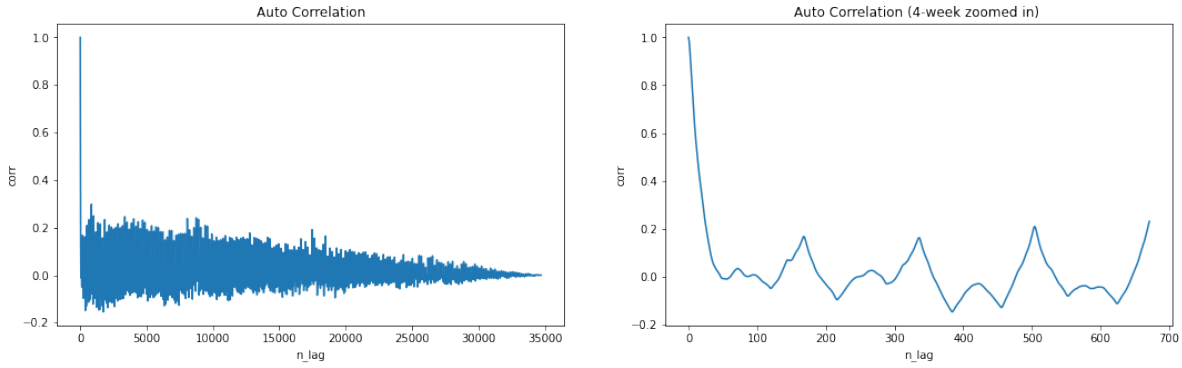
```
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/tsatools.py:142: FutureWarning: In a f
  x = pd.concat(x[::order], 1)
```

Table 6.12: OLS Regression Results

Dep. Variable:	power	R-squared:	0.848
Model:	OLS	Adj. R-squared:	0.847
Method:	Least Squares	F-statistic:	3.860e+04
Date:	Sun, 22 Jan 2023	Prob (F-statistic):	0.00
Time:	19:25:04	Log-Likelihood:	-2.5626e+05
No. Observations:	34728	AIC:	5.125e+05
Df Residuals:	34722	BIC:	5.126e+05
Df Model:	5		
Covariance Type:	nonrobust		

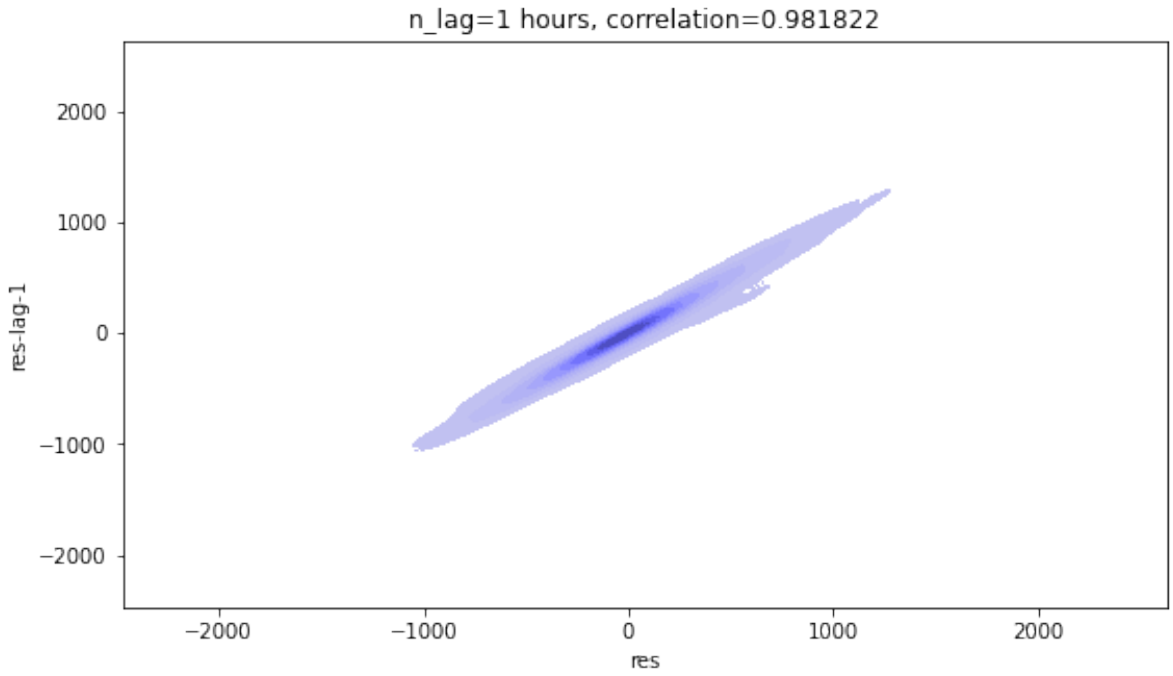
```
plt_acf(res)
```

```
/usr/local/lib/python3.8/dist-packages/statsmodels/tsa/stattools.py:667: FutureWarning: fft=T
  warnings.warn(
```

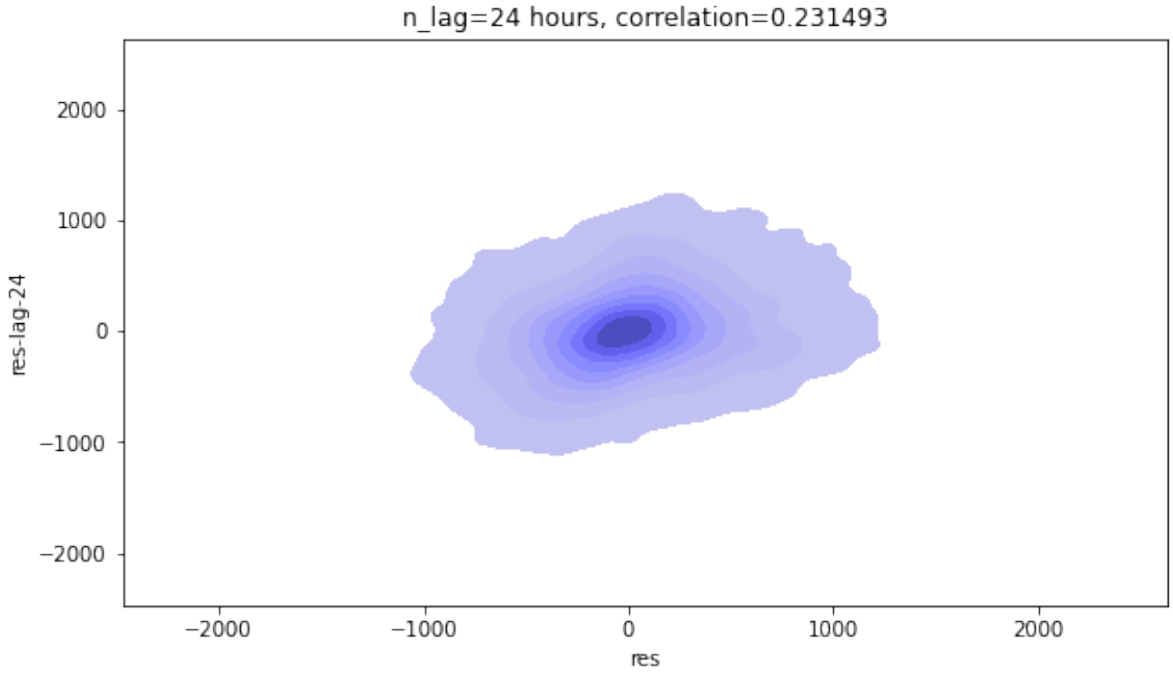


day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1.00	0.98	0.94	0.90	0.85	0.80	0.75	0.71	0.67	0.63	0.59	0.56	0.53	0.50	0.4
1	0.23	0.21	0.20	0.18	0.16	0.14	0.13	0.11	0.10	0.08	0.07	0.06	0.05	0.05	0.0
2	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	-0.00	-0.00	0.0
3	0.03	0.03	0.03	0.03	0.02	0.02	0.02	0.01	0.01	0.01	0.00	0.00	0.00	-0.00	-0
4	0.01	0.01	0.00	0.00	0.00	-0.00	-0.00	-0.01	-0.01	-0.01	-0.02	-0.02	-0.02	-0.02	-0
5	-0.05	-0.05	-0.04	-0.04	-0.04	-0.03	-0.03	-0.03	-0.02	-0.02	-0.02	-0.01	-0.01	-0.00	0.0
6	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.08	0.08	0.09	0.09	0.10	0.10	0.1
7	0.17	0.16	0.15	0.14	0.13	0.12	0.11	0.10	0.09	0.09	0.08	0.07	0.07	0.06	0.0
8	-0.00	-0.01	-0.01	-0.02	-0.02	-0.02	-0.03	-0.03	-0.03	-0.04	-0.04	-0.04	-0.05	-0.05	-0
9	-0.10	-0.10	-0.09	-0.09	-0.09	-0.08	-0.08	-0.08	-0.07	-0.07	-0.06	-0.06	-0.06	-0.05	-0
10	-0.01	-0.01	-0.01	-0.01	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	0.00	0.00	0.00	0.00	0.0
11	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.01	0.01	0.01	0.01	0.0
12	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0.02	-0.02	-0.02	-0.02	-0.01	-0.01	-0
13	0.05	0.05	0.05	0.05	0.06	0.06	0.06	0.07	0.07	0.08	0.08	0.09	0.09	0.10	0.1
14	0.16	0.16	0.15	0.14	0.13	0.12	0.11	0.10	0.09	0.08	0.08	0.07	0.06	0.06	0.0
15	-0.02	-0.03	-0.03	-0.04	-0.05	-0.05	-0.06	-0.06	-0.07	-0.08	-0.08	-0.09	-0.09	-0.10	-0
16	-0.15	-0.15	-0.14	-0.14	-0.13	-0.13	-0.12	-0.12	-0.11	-0.11	-0.11	-0.10	-0.10	-0.10	-0
17	-0.05	-0.05	-0.05	-0.04	-0.04	-0.04	-0.04	-0.04	-0.03	-0.03	-0.03	-0.03	-0.03	-0.03	-0
18	-0.05	-0.05	-0.06	-0.06	-0.06	-0.07	-0.07	-0.07	-0.07	-0.08	-0.08	-0.09	-0.09	-0.09	-0
19	-0.13	-0.13	-0.12	-0.11	-0.11	-0.10	-0.09	-0.09	-0.08	-0.08	-0.07	-0.07	-0.06	-0.05	-0
20	0.02	0.03	0.03	0.04	0.04	0.05	0.05	0.06	0.07	0.07	0.08	0.09	0.10	0.10	0.1
21	0.21	0.20	0.19	0.18	0.17	0.16	0.15	0.13	0.12	0.11	0.11	0.10	0.09	0.09	0.0
22	0.03	0.03	0.02	0.02	0.01	0.00	-0.00	-0.01	-0.01	-0.02	-0.02	-0.03	-0.03	-0.04	-0
23	-0.08	-0.08	-0.08	-0.08	-0.07	-0.07	-0.07	-0.06	-0.06	-0.06	-0.06	-0.05	-0.05	-0.05	-0
24	-0.04	-0.04	-0.04	-0.04	-0.04	-0.04	-0.05	-0.05	-0.05	-0.05	-0.05	-0.05	-0.05	-0.05	-0
25	-0.04	-0.04	-0.05	-0.05	-0.05	-0.05	-0.05	-0.06	-0.06	-0.06	-0.07	-0.07	-0.07	-0.08	-0
26	-0.11	-0.11	-0.11	-0.10	-0.10	-0.09	-0.09	-0.08	-0.08	-0.08	-0.07	-0.07	-0.06	-0.06	-0
27	0.02	0.02	0.03	0.04	0.04	0.05	0.06	0.07	0.07	0.09	0.10	0.11	0.11	0.12	0.1
28	0.23	0.23	0.22	0.20	0.19	0.18	0.17	0.16	0.15	0.14	0.13	0.12	0.11	0.11	0.1
29	0.03	0.03	0.02	0.01	0.01	-0.00	-0.01	-0.01	-0.02	-0.03	-0.03	-0.04	-0.05	-0.05	-0

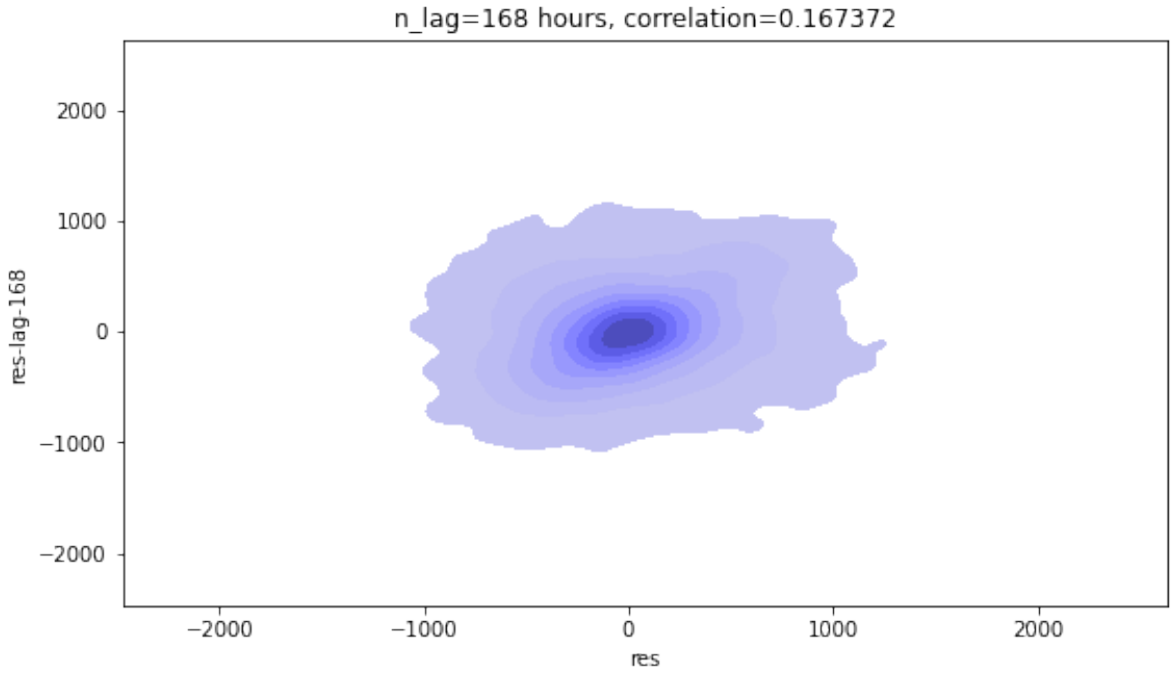
```
plt_residual_lag(res, 1)
```



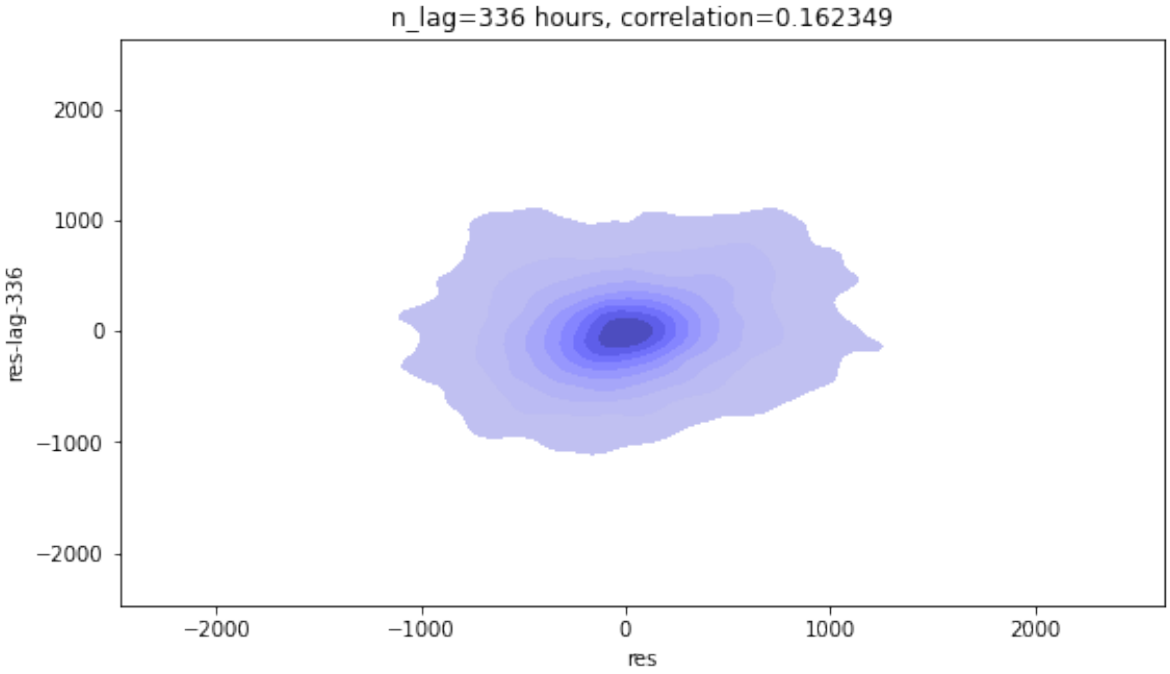

```
plt_residual_lag(res, 24)
```



```
plt_residual_lag(res, 24*7)
```




```
plt_residual_lag(res, 24*7*2)
```


n_lag=336 hours, correlation=0.162349
2000
1000
0
−1000
−2000
res-lag-336
−2000
−1000
0
1000
2000
res

7 Remark

We saw that with 2-week-lag feature, the R^2 only increased a little. The model summary seems still good so we could keep it. However, from the viewpoint of interpretation I may remove it.

One may also notice that the 1-day-lag correlation becomes bigger although 1-day-lag feature is already in the model. It is probably because of the multicollinearity between the lag features.

The following table shows the correlation between lag features.

```
df[['power_lag_1_day','power_lag_1_week', 'power_lag_2_week' ]].corr()
```

	power_lag_1_day	power_lag_1_week	power_lag_2_week
power_lag_1_day	1.000000	0.768394	0.745817
power_lag_1_week	0.768394	1.000000	0.819955
power_lag_2_week	0.745817	0.819955	1.000000

Part II

Logistic regression

8 Logistic regression

Read sections 4.1 - 4.3 of the book before using these notes.

Note that in this course, lecture notes are not sufficient, you must read the book for better understanding. Lecture notes are just implementing the concepts of the book on a dataset, but not explaining the concepts elaborately.

Logistic regression is the go-to linear classification algorithm for two-class problems. It is easy to implement, easy to understand and gets great results on a wide variety of problems, even when the expectations the method has for your data are violated.

8.0.1 Description

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the **`Sigmoid function`** was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

$$\frac{1}{1+e^{-x}}$$

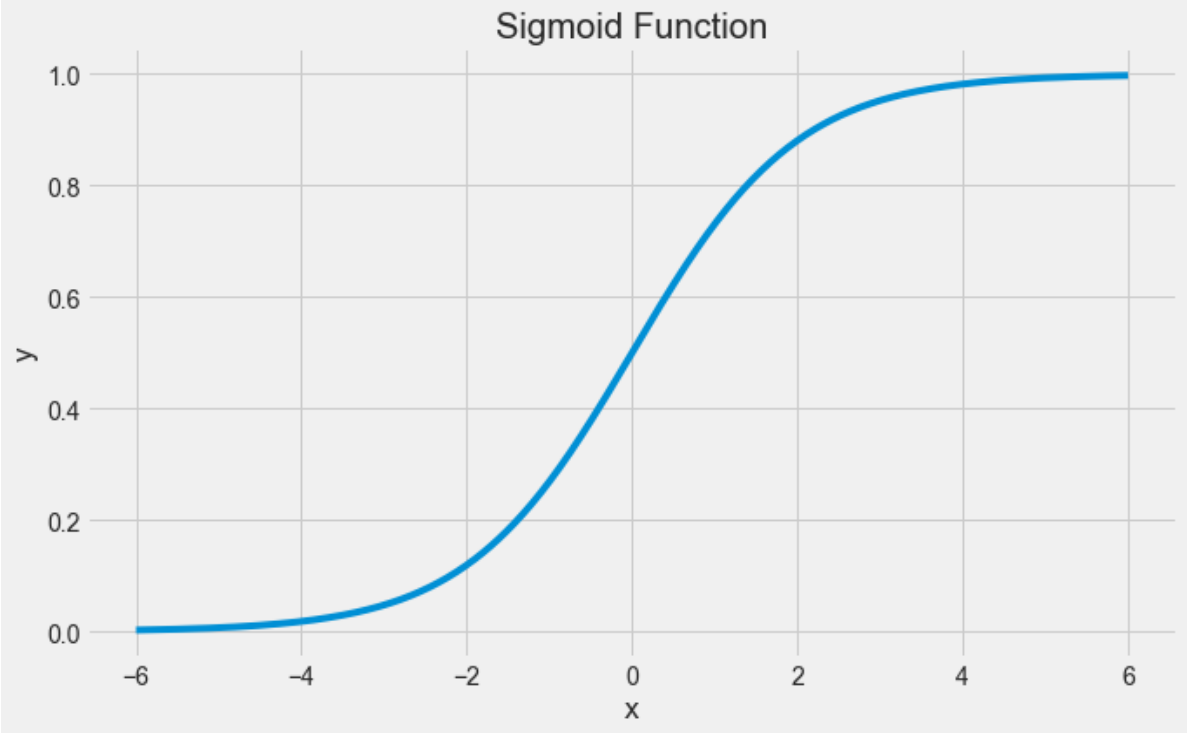
e is the base of the natural logarithms and x is value that you want to transform via the logistic function.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.formula.api as sm
```

```
%matplotlib inline
sns.set_style('whitegrid')
plt.style.use("fivethirtyeight")
```

```
x = np.linspace(-6, 6, num=1000)
plt.figure(figsize=(10, 6))
plt.plot(x, (1 / (1 + np.exp(-x))))
plt.xlabel("x")
plt.ylabel("y")
plt.title("Sigmoid Function")
```

```
Text(0.5, 1.0, 'Sigmoid Function')
```



The logistic regression equation has a very similar representation like linear regression. The difference is that the output value being modelled is binary in nature.

$$\hat{p} = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 x_1}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 x_1}}$$

or

$$\hat{p} = \frac{1.0}{1.0 + e^{-(\hat{\beta}_0 + \hat{\beta}_1 x_1)}}$$

$\hat{\beta}_0$ is the estimated intercept term

$\hat{\beta}_1$ is the estimated coefficient for x_1

$\hat{p}$ is the predicted output with real value between 0 and 1. To convert this to binary output of 0 or 1, this would either need to be rounded to an integer value or a cutoff point be provided to specify the class segregation point.

8.0.2 Learning the Logistic Regression Model

The coefficients (Beta values b) of the logistic regression algorithm must be estimated from your training data. This is done using maximum-likelihood estimation.

Maximum-likelihood estimation is a common learning algorithm used by a variety of machine learning algorithms, although it does make assumptions about the distribution of your data (more on this when we talk about preparing your data).

The best coefficients should result in a model that would predict a value very close to 1 (e.g. male) for the default class and a value very close to 0 (e.g. female) for the other class. The intuition for maximum-likelihood for logistic regression is that a search procedure seeks values for the coefficients (Beta values) that maximize the likelihood of the observed data. In other words, in MLE, we estimate the parameter values (Beta values) which are the most likely to produce that data at hand.

Here is an analogy to understand the idea behind Maximum Likelihood Estimation (MLE). Let us say, you are listening to a song (data). You are not aware of the singer (parameter) of the song. With just the musical piece at hand, you try to guess the singer (parameter) who you feel is the most likely (MLE) to have sung that song. Your are making a maximum likelihood estimate! Out of all the singers (parameter space) you have chosen them as the one who is the most likely to have sung that song (data).

We are not going to go into the math of maximum likelihood. It is enough to say that a minimization algorithm is used to optimize the best values for the coefficients for your training data. This is often implemented in practice using efficient numerical optimization algorithm (like the Quasi-newton method).

When you are learning logistic, you can implement it yourself from scratch using the much simpler gradient descent algorithm.

8.0.3 Preparing Data for Logistic Regression

The assumptions made by logistic regression about the distribution and relationships in your data are much the same as the assumptions made in linear regression.

Much study has gone into defining these assumptions and precise probabilistic and statistical language is used. My advice is to use these as guidelines or rules of thumb and experiment with different data preparation schemes.

Ultimately in predictive modeling machine learning projects you are laser focused on making accurate predictions rather than interpreting the results. As such, you can break some assumptions as long as the model is robust and performs well.

- **Binary Output Variable:** This might be obvious as we have already mentioned it, but logistic regression is intended for binary (two-class) classification problems. It will predict the probability of an instance belonging to the default class, which can be snapped into a 0 or 1 classification.
- **Remove Noise:** Logistic regression assumes no error in the output variable (y), consider removing outliers and possibly misclassified instances from your training data.
- **Gaussian Distribution:** Logistic regression is a linear algorithm (with a non-linear transform on output). It does assume a linear relationship between the input variables with the output. Data transforms of your input variables that better expose this linear relationship can result in a more accurate model. For example, you can use log, root, Box-Cox and other univariate transforms to better expose this relationship.
- **Remove Correlated Inputs:** Like linear regression, the model can overfit if you have multiple highly-correlated inputs. Consider calculating the pairwise correlations between all inputs and removing highly correlated inputs.
- **Fail to Converge:** It is possible for the expected likelihood estimation process that learns the coefficients to fail to converge. This can happen if there are many highly correlated inputs in your data or the data is very sparse (e.g. lots of zeros in your input data).

8.1 Logistic Regression: Scikit-learn vs Statsmodels

Python gives us two ways to do logistic regression. Statsmodels offers modeling from the perspective of statistics. Scikit-learn offers some of the same models from the perspective of machine learning.

So we need to understand the difference between statistics and machine learning! Statistics makes mathematically valid inferences about a population based on sample data. Statistics answers the question, "What is the evidence that X is related to Y?" Machine learning has the goal of optimizing predictive accuracy rather than inference. Machine learning answers the question, "Given X, what prediction should we make for Y?"

Let us see the use of `statsmodels` for logistic regression. We'll see scikit-learn later in the course, when we learn methods that focus on prediction.

8.2 Training a logistic regression model

Read the data on social network ads. The data shows if the person purchased a product when targeted with an ad on social media. Fit a logistic regression model to predict if a user will purchase the product based on their characteristics such as age, gender and estimated salary.

```
train = pd.read_csv('./Datasets/Social_Network_Ads_train.csv') #Develop the model on train
test = pd.read_csv('./Datasets/Social_Network_Ads_test.csv') #Test the model on test data
```

```
train.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15755018	Male	36	33000	0
1	15697020	Female	39	61000	0
2	15796351	Male	36	118000	1
3	15665760	Male	39	122000	1
4	15794661	Female	26	118000	0

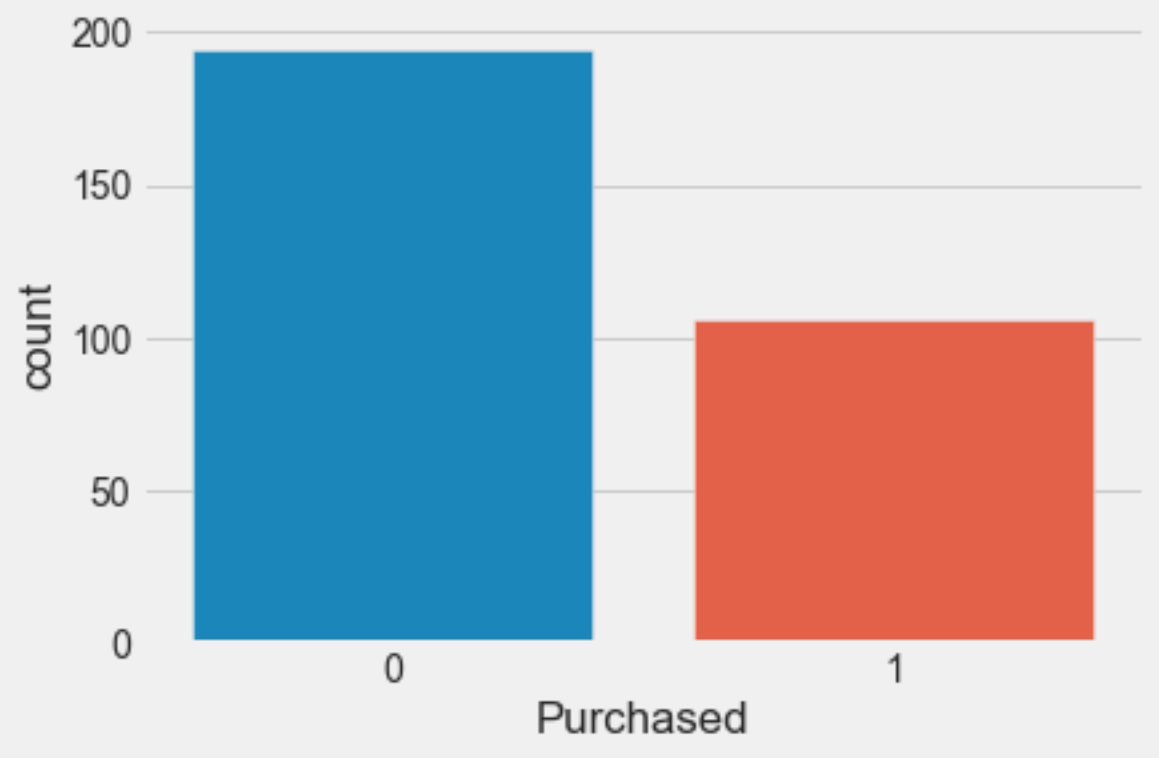
8.2.1 Examining the Distribution of the Target Column

Make sure our target is not severely imbalanced.

```
train.Purchased.value_counts()
```

```
0    194
1    106
Name: Purchased, dtype: int64
```

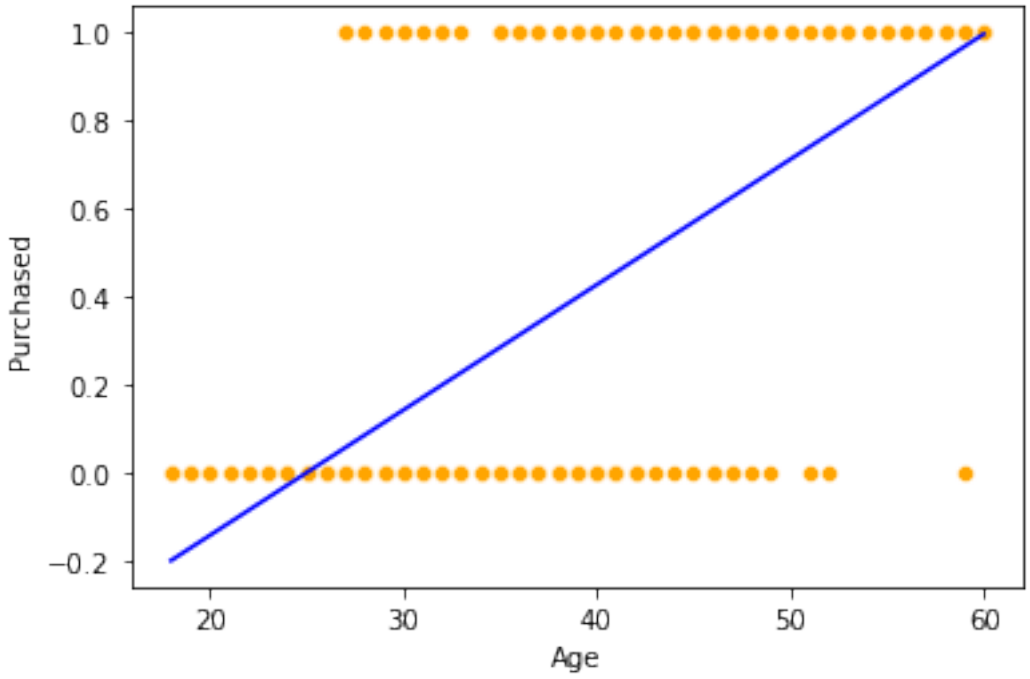
```
sns.countplot(x = 'Purchased',data = train);
```



Let us try to fit a linear regression model, instead of logistic regression. We fit a linear regression model to predict probability of purchase based on age.

```
sns.scatterplot(x = 'Age', y = 'Purchased', data = train, color = 'orange') #Visualizing d
lm = sm.ols(formula = 'Purchased~Age', data = train).fit() #Developing linear regression m
sns.lineplot(x = 'Age', y= lm.predict(train), data = train, color = 'blue') #Visualizing m
```

```
<AxesSubplot:xlabel='Age', ylabel='Purchased'>
```



Note the issues with the linear regression model:

1. The regression line goes below 0 and over 1. However, probability of purchase must be in [0,1].
2. The linear regression model does not seem to fit the data well.

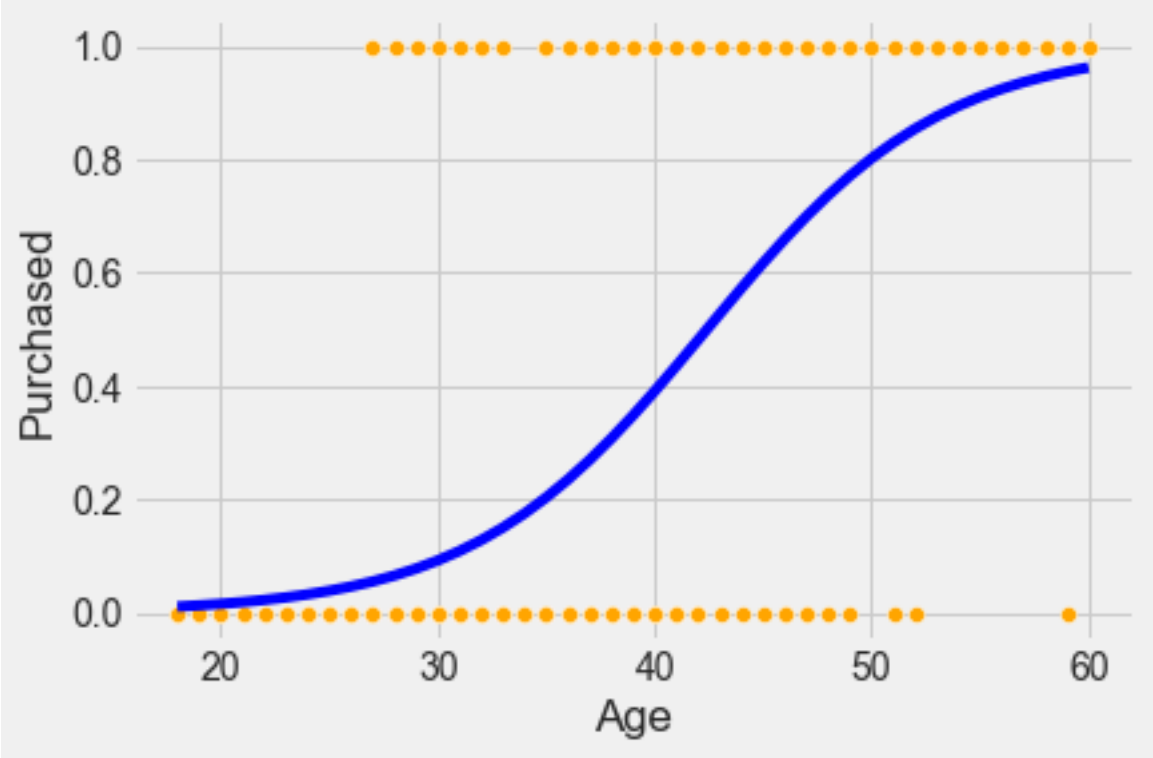
8.2.2 Fitting the logistic regression model

Now, let us fit a logistic regression model to predict probability of purchase based on `Age`.

```
sns.scatterplot(x = 'Age', y = 'Purchased', data = train, color = 'orange') #Visualizing d
logit_model = sm.logit(formula = 'Purchased~Age', data = train).fit() #Developing logistic
sns.lineplot(x = 'Age', y= logit_model.predict(train), data = train, color = 'blue') #Visu
```

```
Optimization terminated successfully.
         Current function value: 0.430107
         Iterations 7
```

```
<AxesSubplot:xlabel='Age', ylabel='Purchased'>
```



As logistic regression uses the sigmoid function, the probability stays in [0,1]. Also, it seems to better fit the points as compared to linear regression.

```
logit_model.summary()
```

Table 8.2: Logit Regression Results

Dep. Variable:	Purchased	No. Observations:	300
Model:	Logit	Df Residuals:	298
Method:	MLE	Df Model:	1
Date:	Tue, 19 Apr 2022	Pseudo R-squ.:	0.3378
Time:	16:46:02	Log-Likelihood:	-129.03
converged:	True	LL-Null:	-194.85
Covariance Type:	nonrobust	LLR p-value:	1.805e-30

Interpret the coefficient of age

For a unit increase in age, the log odds of purchase increase by 0.18, or the odds of purchase get multiplied by exp(0.18) = 1.2

Is the increase in probability of purchase constant with a unit increase in age?

No, it depends on age.

Is gender associated with probability of purchase?

```
logit_model_gender = sm.logit(formula = 'Purchased~Gender', data = train).fit()
logit_model_gender.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.648804
         Iterations 4
```

Table 8.3: Logit Regression Results

Dep. Variable:	Purchased	No. Observations:	300
Model:	Logit	Df Residuals:	298
Method:	MLE	Df Model:	1
Date:	Tue, 19 Apr 2022	Pseudo R-squ.:	0.001049
Time:	16:46:04	Log-Likelihood:	-194.64
converged:	True	LL-Null:	-194.85
Covariance Type:	nonrobust	LLR p-value:	0.5225

No, assuming a significance level of $\alpha = 5\%$, `Gender` is not associated with probability of default, as the p-value for `Male` is greater than 0.05.

8.3 Confusion matrix and classification accuracy

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class.

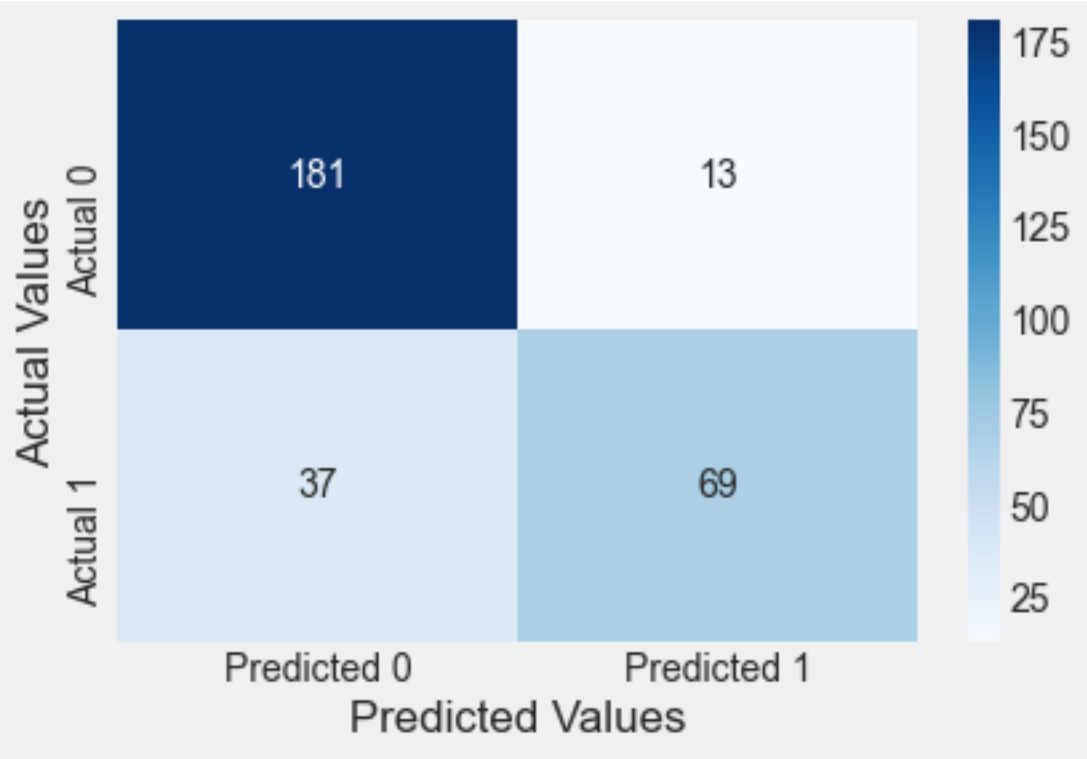
```
#Function to compute confusion matrix and prediction accuracy on training data
def confusion_matrix_train(model,cutoff=0.5):
    # Confusion matrix
    cm_df = pd.DataFrame(model.pred_table(threshold = cutoff))
    #Formatting the confusion matrix
    cm_df.columns = ['Predicted 0', 'Predicted 1']
    cm_df = cm_df.rename(index={0: 'Actual 0',1: 'Actual 1'})
    cm = np.array(cm_df)
```

```
    # Calculate the accuracy
    accuracy = (cm[0,0]+cm[1,1])/cm.sum()
    sns.heatmap(cm_df, annot=True, cmap='Blues', fmt='g')
    plt.ylabel("Actual Values")
    plt.xlabel("Predicted Values")
    print("Classification accuracy = {:.1%}".format(accuracy))
```

Find the confusion matrix and classification accuracy of the model with `Age` as the predictor on training data.

```
cm = confusion_matrix_train(logit_model)
```

```
Classification accuracy = 83.3%
```



Confusion matrix:

- Each row: actual class
- Each column: predicted class

First row: Non-purchasers, the negative class:

- 181 were correctly classified as Non-purchasers. **True negatives**.
- Remaining 13 were wrongly classified as Non-purchasers. **False positive**

Second row: Purchasers, the positive class:

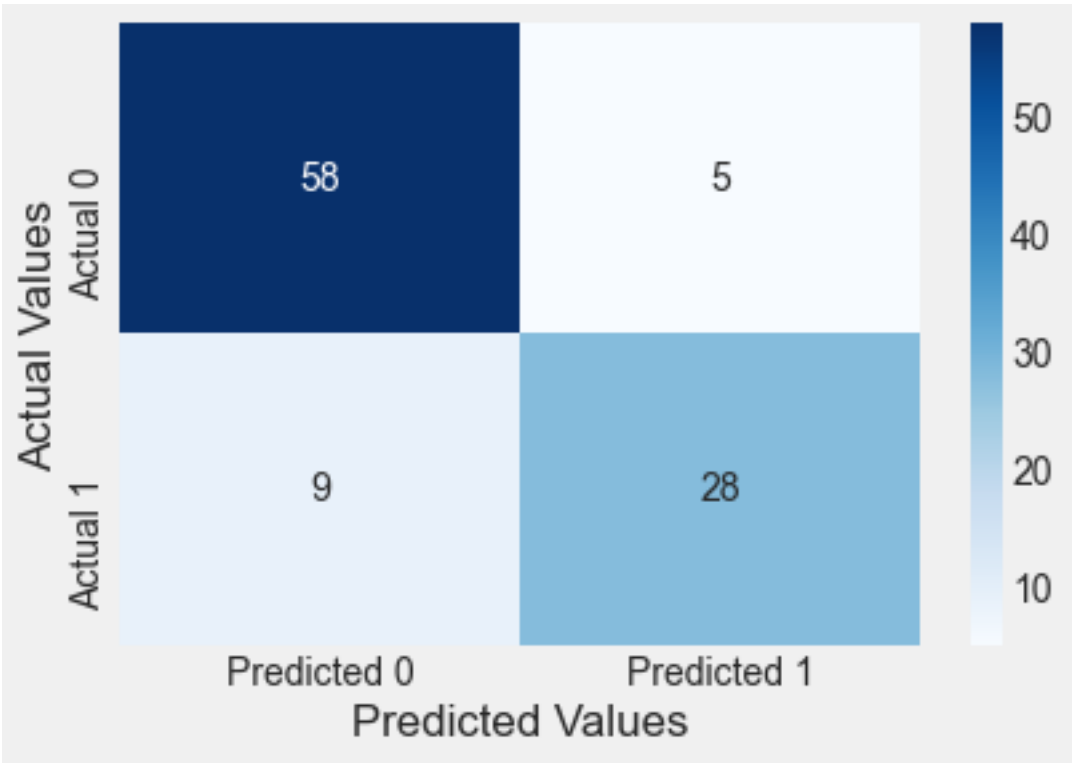
- 37 were incorrectly classified as Non-purchasers. **False negatives**
- 69 were correctly classified Purchasers. **True positives**

```
#Function to compute confusion matrix and prediction accuracy on test data
def confusion_matrix_test(data,actual_values,model,cutoff=0.5):
#Predict the values using the Logit model
    pred_values = model.predict(data)
# Specify the bins
    bins=np.array([0,cutoff,1])
#Confusion matrix
    cm = np.histogram2d(actual_values, pred_values, bins=bins)[0]
    cm_df = pd.DataFrame(cm)
    cm_df.columns = ['Predicted 0','Predicted 1']
    cm_df = cm_df.rename(index={0: 'Actual 0',1:'Actual 1'})
    accuracy = (cm[0,0]+cm[1,1])/cm.sum()
    sns.heatmap(cm_df, annot=True, cmap='Blues', fmt='g')
    plt.ylabel("Actual Values")
    plt.xlabel("Predicted Values")
    print("Classification accuracy = {:.1%}".format(accuracy))
```

Find the confusion matrix and classification accuracy of the model with `Age` as the predictor on test data.

```
confusion_matrix_test(test,test.Purchased,logit_model)
```

```
Classification accuracy = 86.0%
```



The model classifies a bit more accurately on test data as compared to the training data, which is a bit unusual. However, it shows that the model did not overfit on training data.

Include `EstimatedSalary` as a predictor in the above model

```
logit_model2 = sm.logit(formula = 'Purchased~Age+EstimatedSalary', data = train).fit()
logit_model2.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.358910
         Iterations 7
```

Table 8.4: Logit Regression Results

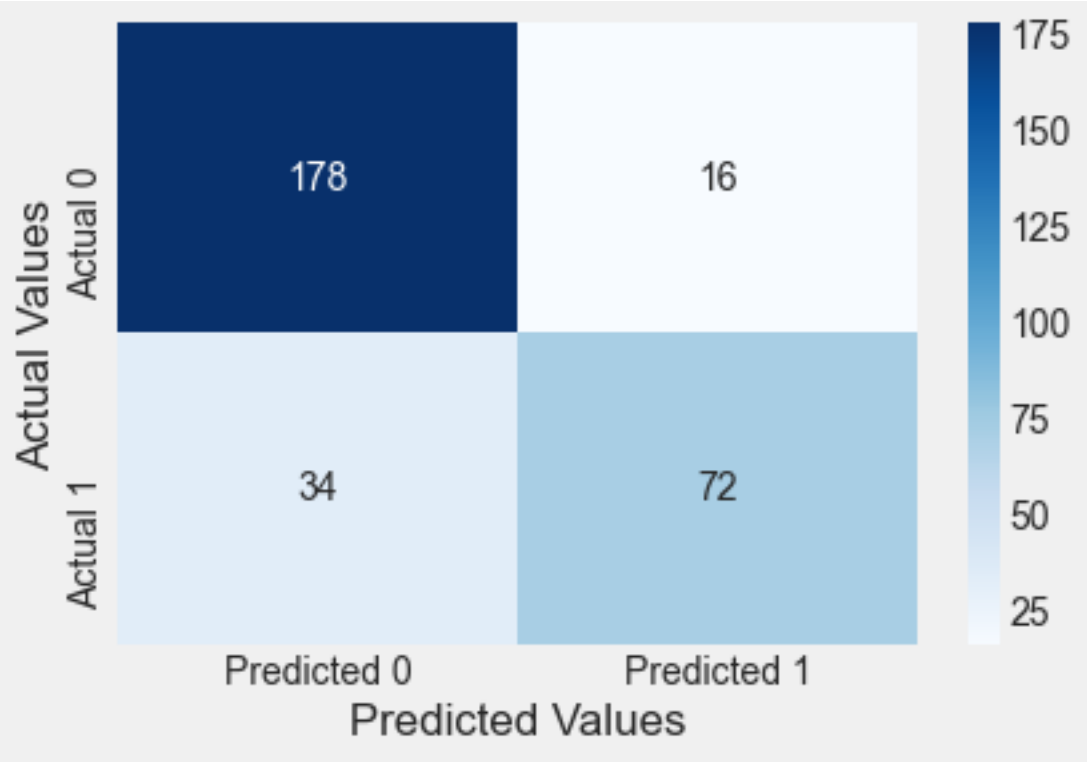
Dep. Variable:	Purchased	No. Observations:	300
Model:	Logit	Df Residuals:	297
Method:	MLE	Df Model:	2
Date:	Tue, 14 Feb 2023	Pseudo R-squ.:	0.4474
Time:	12:03:29	Log-Likelihood:	-107.67

Table 8.4: Logit Regression Results

converged:	True	LL-Null:	-194.85
Covariance Type:	nonrobust	LLR p-value:	1.385e-38

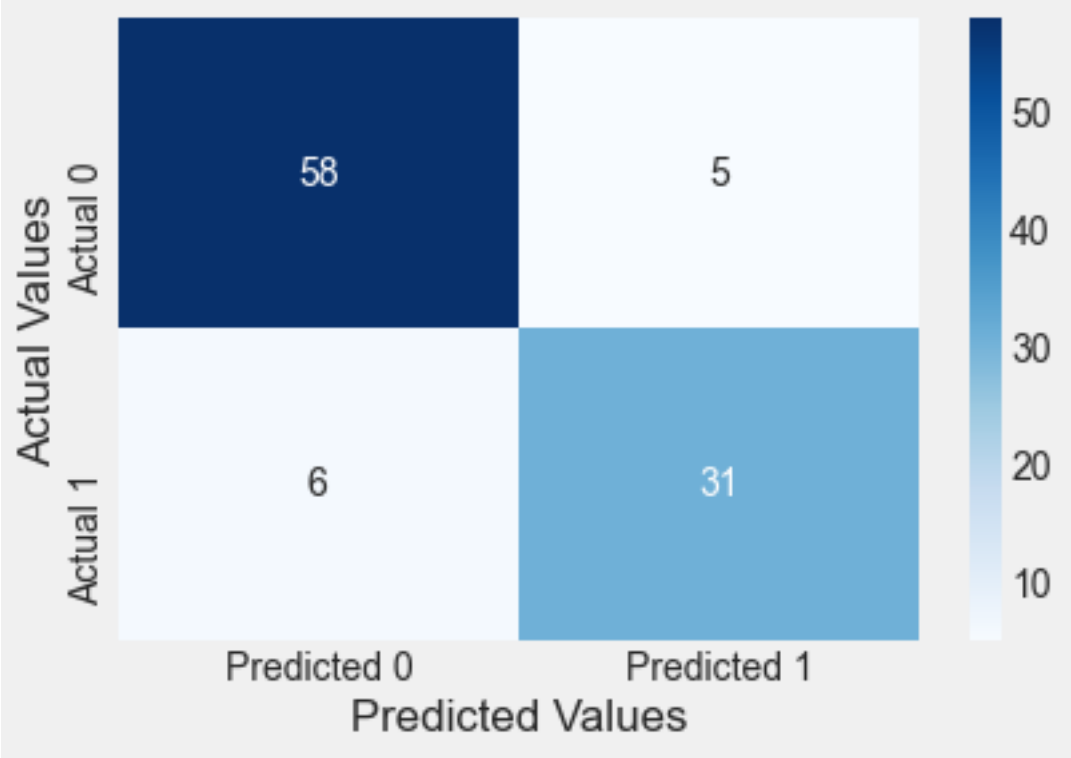
```
confusion_matrix_train(logit_model2)
```

```
Classification accuracy = 83.3%
```




```
confusion_matrix_test(test,test.Purchased,logit_model2)
```

```
Classification accuracy = 89.0%
```



The log likelihood of the model has increased, while also increasing the prediction accuracy on test data, which shows that the additional predictor is helping explain the response better, without overfitting the data.

Include `Gender` as a predictor in the above model

```
logit_model = sm.logit(formula = 'Purchased~Age+EstimatedSalary+Gender', data = train).fit
logit_model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.357327
         Iterations 7
```

Table 8.5: Logit Regression Results

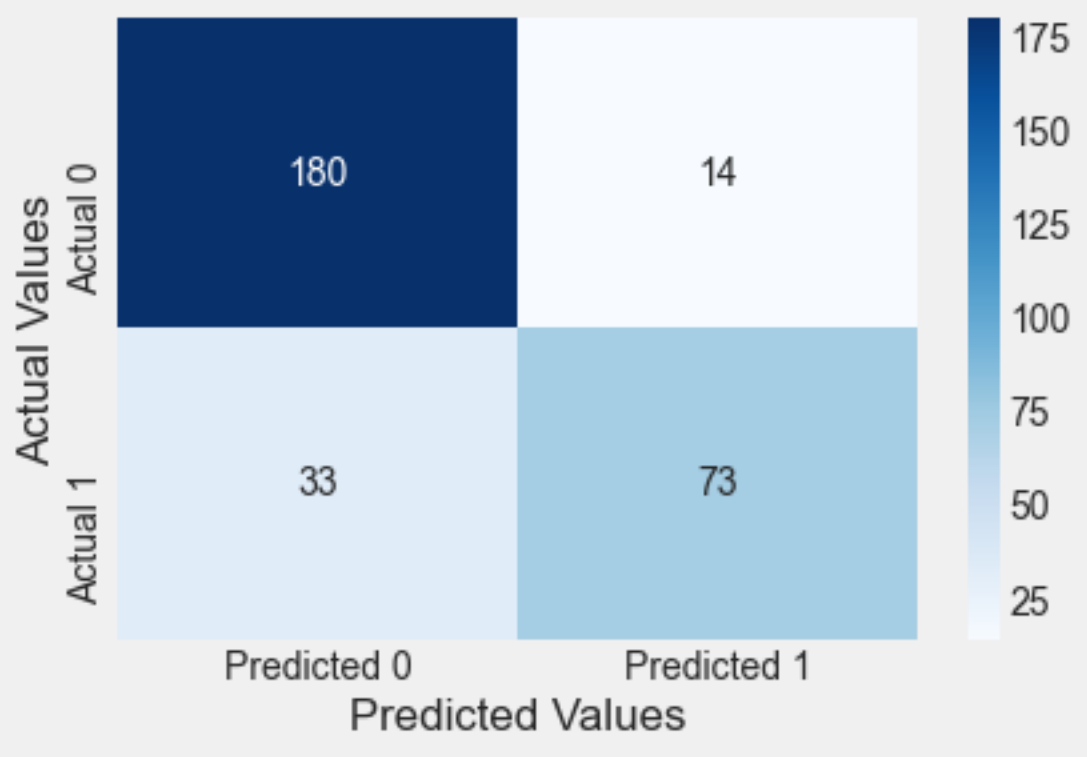
Dep. Variable:	Purchased	No. Observations:	300
Model:	Logit	Df Residuals:	296
Method:	MLE	Df Model:	3
Date:	Tue, 14 Feb 2023	Pseudo R-squ.:	0.4498

Table 8.5: Logit Regression Results

Time:	12:17:28	Log-Likelihood:	-107.20
converged:	True	LL-Null:	-194.85
Covariance Type:	nonrobust	LLR p-value:	9.150e-38

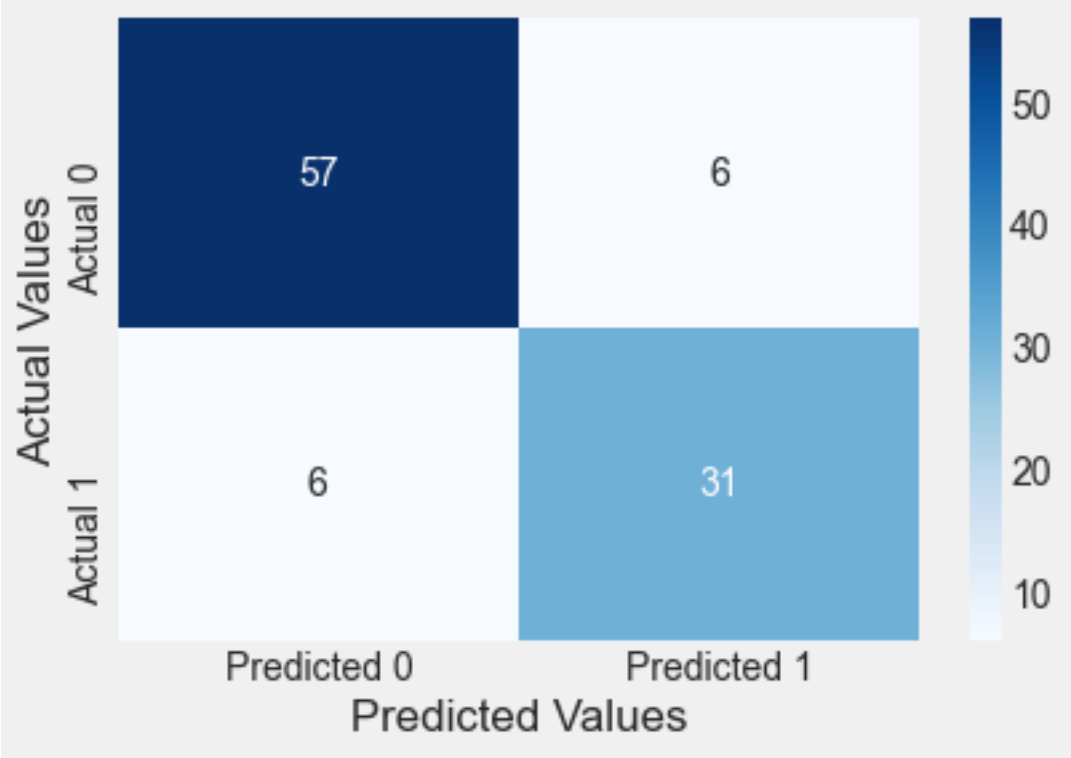
```
confusion_matrix_train(logit_model)
```

```
Classification accuracy = 84.3%
```




```
confusion_matrix_test(test,test.Purchased,logit_model)
```

```
Classification accuracy = 88.0%
```



`Gender` is a statistically insignificant predictor, and including it slightly lowers the classification accuracy on test data. Note that the classification accuracy on training data will continue to increase on adding more predictors, irrespective of their relevance *(similar to the idea of RSS on training data in linear regression)*.

Is there a residual in logistic regression?

No, since the response is assumed to have a Bernoulli distribution, instead of a normal distribution.

Is the odds ratio for a unit increase in a predictor X_j, a constant (assuming that the rest of the predictors are held constant)?

Yes, the odds ratio in this case will e^{β_j}

8.4 Variable transformations in logistic regression

Read the dataset *diabetes.csv* that contains if a person has diabetes (`Outcome = 1`) based on health parameters such as BMI, blood pressure, age etc. Develop a model to predict the probability of a person having diabetes based on their age.

```
data = pd.read_csv('./Datasets/diabetes.csv')
```

```
data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33

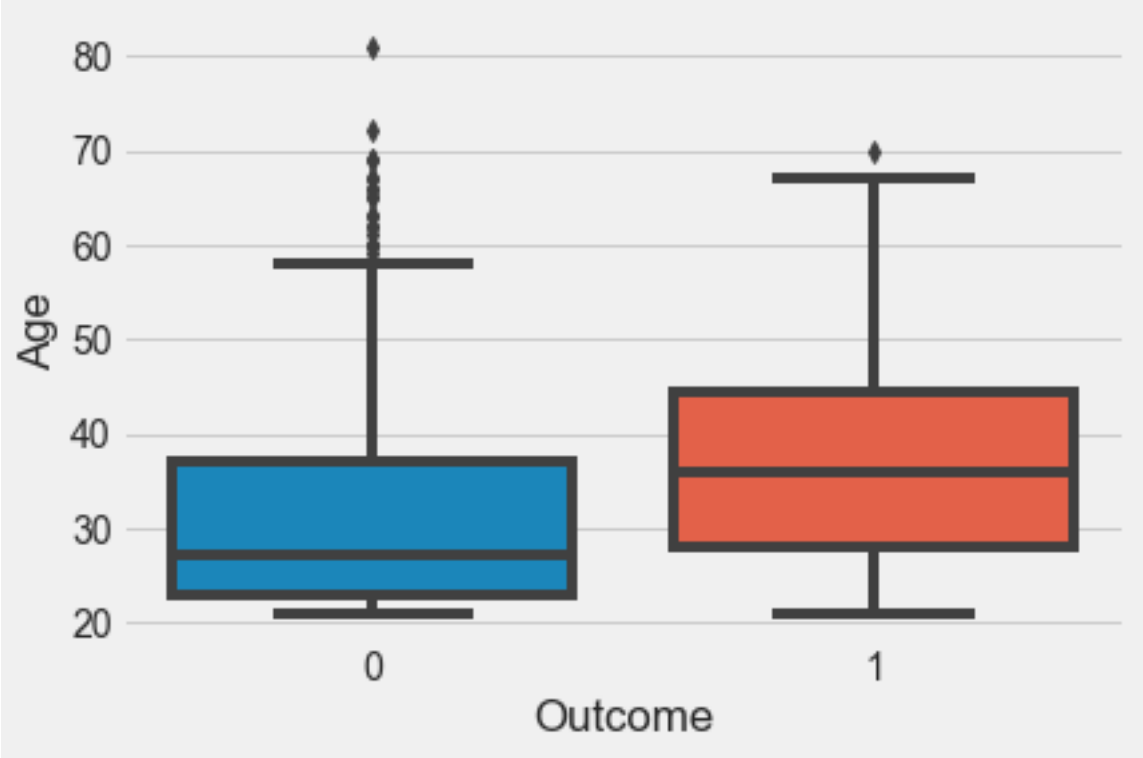
Randomly select 80% of the observations to create a training dataset. Create a test dataset with the remaining 20% observations.

```
#Creating training and test datasets
np.random.seed(2)
train = data.sample(round(data.shape[0]*0.8))
test = data.drop(train.index)
```

Does `Age` seem to distinguish `Outcome` levels?

```
sns.boxplot(x = 'Outcome', y = 'Age', data = train)
```

```
<AxesSubplot:xlabel='Outcome', ylabel='Age'>
```

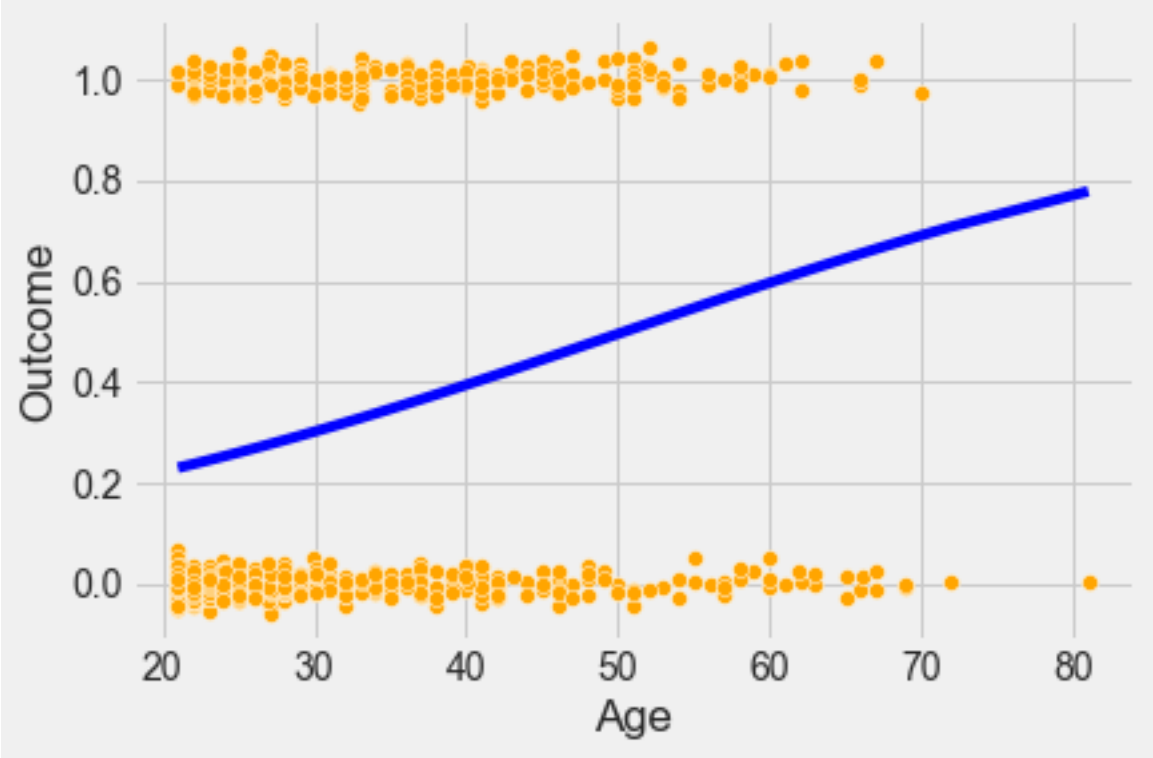


Yes it does!

Develop and visualize a logistic regression model to predict `Outcome` using `Age`.

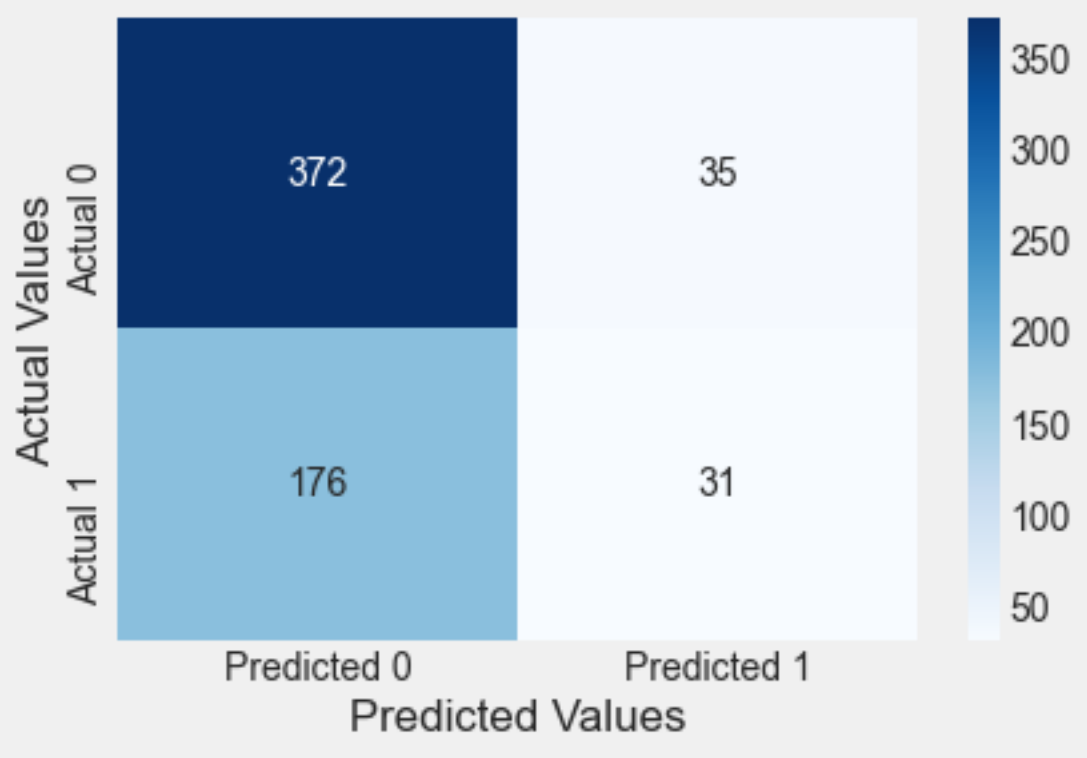
```
#Jittering points to better see the density of points in any given region of the plot
def jitter(values,j):
    return values + np.random.normal(j,0.02,values.shape)
sns.scatterplot(x = jitter(train.Age,0), y = jitter(train.Outcome,0), data = train, color
logit_model = sm.logit(formula = 'Outcome~Age', data = train).fit()
sns.lineplot(x = 'Age', y= logit_model.predict(train), data = train, color = 'blue')
print(logit_model.llf) #Printing the log likelihood to compare it with the next model we b
```

```
Optimization terminated successfully.
         Current function value: 0.612356
         Iterations 5
-375.9863802089716
```




```
confusion_matrix_train(logit_model)
```

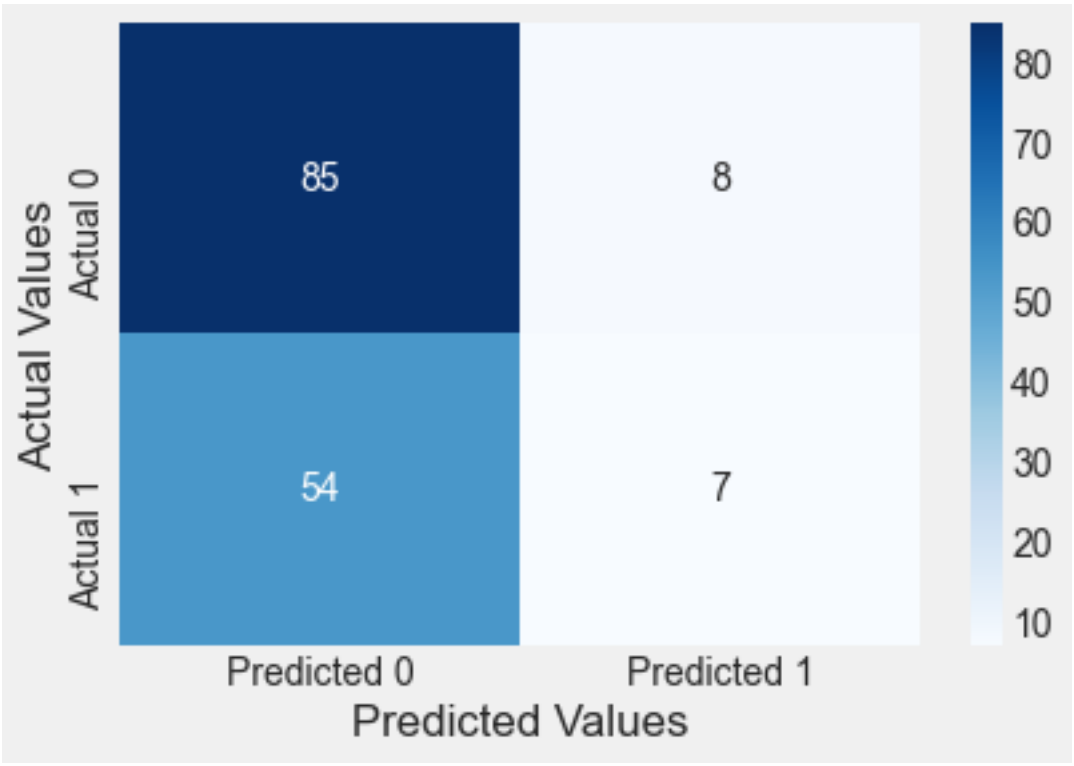
```
Classification accuracy = 65.6%
```



Classification accuracy on train data = 66%

```
confusion_matrix_test(test,test.Outcome,logit_model)
```

```
Classification accuracy = 59.7%
```



Classification accuracy on test data = 60%

Can a tranformation of `Age` provide a more accurate model?

Let us visualize how the probability of people having diabetes varies with `Age`. We will bin `Age` to get the percentage of people having diabetes within different `Age` bins.

```
#Binning Age
binned_age = pd.qcut(train['Age'],11,retbins=True)
train['age_binned'] = binned_age[0]
```

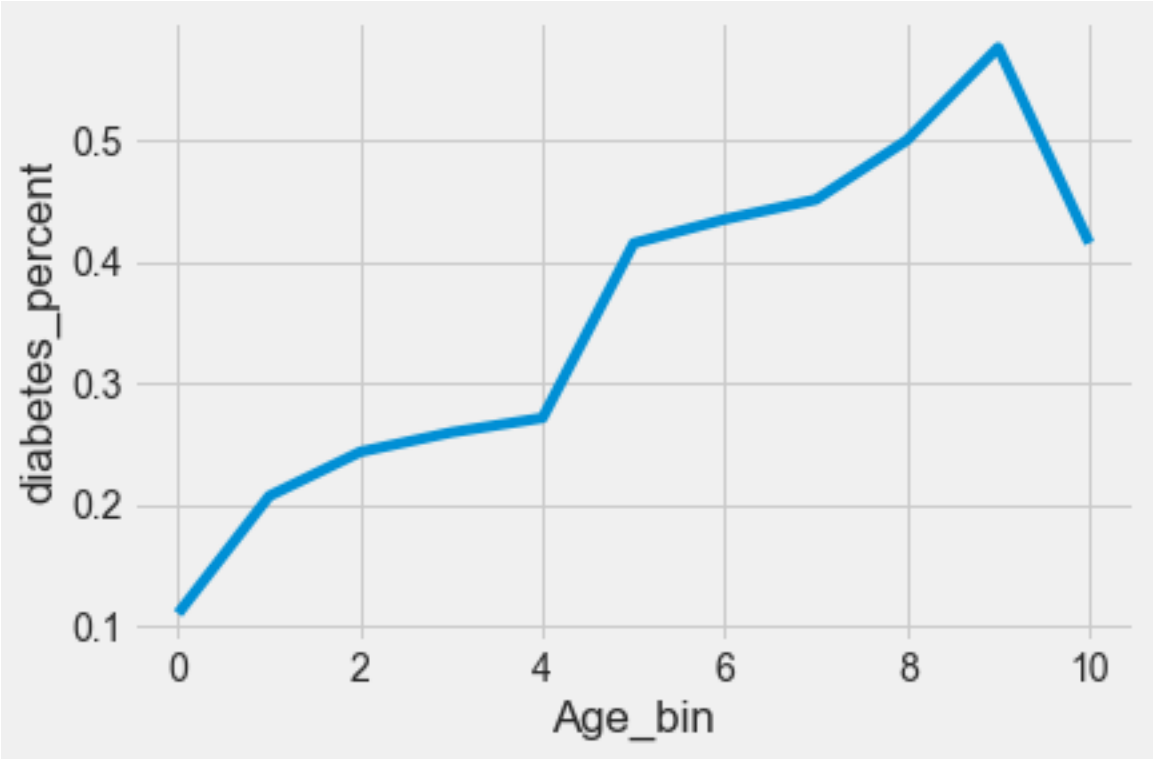
```
#Finding percentage of people having diabetes in each Age bin
age_data = train.groupby('age_binned')['Outcome'].agg([('diabetes_percent','mean'),('nobs'
age_data
```

	age_binned	diabetes_percent	nobs
0	(20.999, 22.0]	0.110092	109
1	(22.0, 23.0]	0.206897	29
2	(23.0, 25.0]	0.243243	74
3	(25.0, 26.0]	0.259259	27

	age_binned	diabetes_percent	nobs
4	(26.0, 28.0]	0.271186	59
5	(28.0, 31.0]	0.415094	53
6	(31.0, 35.0]	0.434783	46
7	(35.0, 39.0]	0.450980	51
8	(39.0, 43.545]	0.500000	54
9	(43.545, 52.0]	0.576271	59
10	(52.0, 81.0]	0.415094	53

```
#Visualizing percentage of people having diabetes with increasing Age (or Age bins)
sns.lineplot(x = age_data.index, y= age_data['diabetes_percent'])
plt.xlabel('Age_bin')
```

```
Text(0.5, 0, 'Age_bin')
```



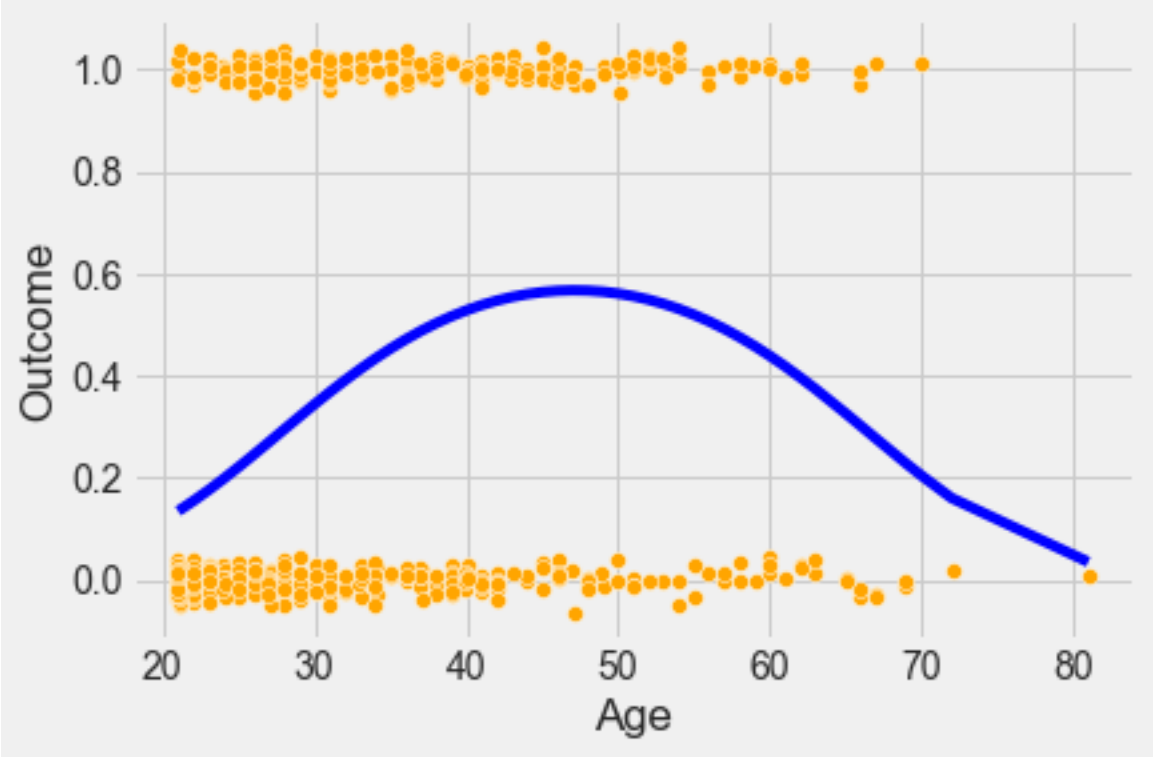
We observe that the probability of people having diabetes does **not** keep increasing monotonically with age. People with ages 52 and more have a lower probability of having diabetes than people in the immediately younger `Age` bin.

A quadratic transformation of `Age` may better fit the above trend

```
#Model with the quadratic transformation of Age
def jitter(values,j):
    return values + np.random.normal(j,0.02,values.shape)
sns.scatterplot(x = jitter(train.Age,0), y = jitter(train.Outcome,0), data = train, color
logit_model = sm.logit(formula = 'Outcome~Age+I(Age**2)', data = train).fit()
sns.lineplot(x = 'Age', y= logit_model.predict(train), data = train, color = 'blue')
logit_model.llf
```

```
Optimization terminated successfully.
         Current function value: 0.586025
         Iterations 6
```

```
-359.81925590230185
```




```
logit_model.summary()
```

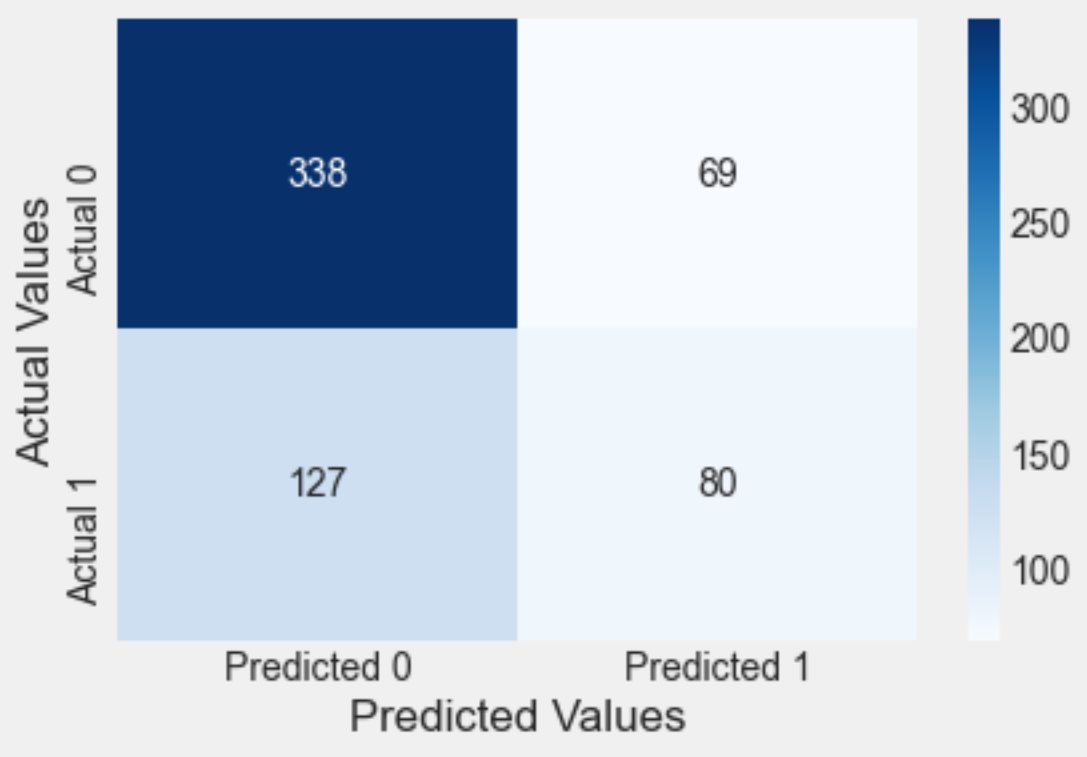
Table 8.8: Logit Regression Results

Dep. Variable:	Outcome	No. Observations:	614
Model:	Logit	Df Residuals:	611
Method:	MLE	Df Model:	2
Date:	Tue, 14 Feb 2023	Pseudo R-squ.:	0.08307
Time:	12:25:54	Log-Likelihood:	-359.82
converged:	True	LL-Null:	-392.42
Covariance Type:	nonrobust	LLR p-value:	6.965e-15

The log likelihood of the model is higher and both the predictors are statistically significant indicating a better model fit. However, the model may also be overfitting. Let us check the model accuracy on test data.

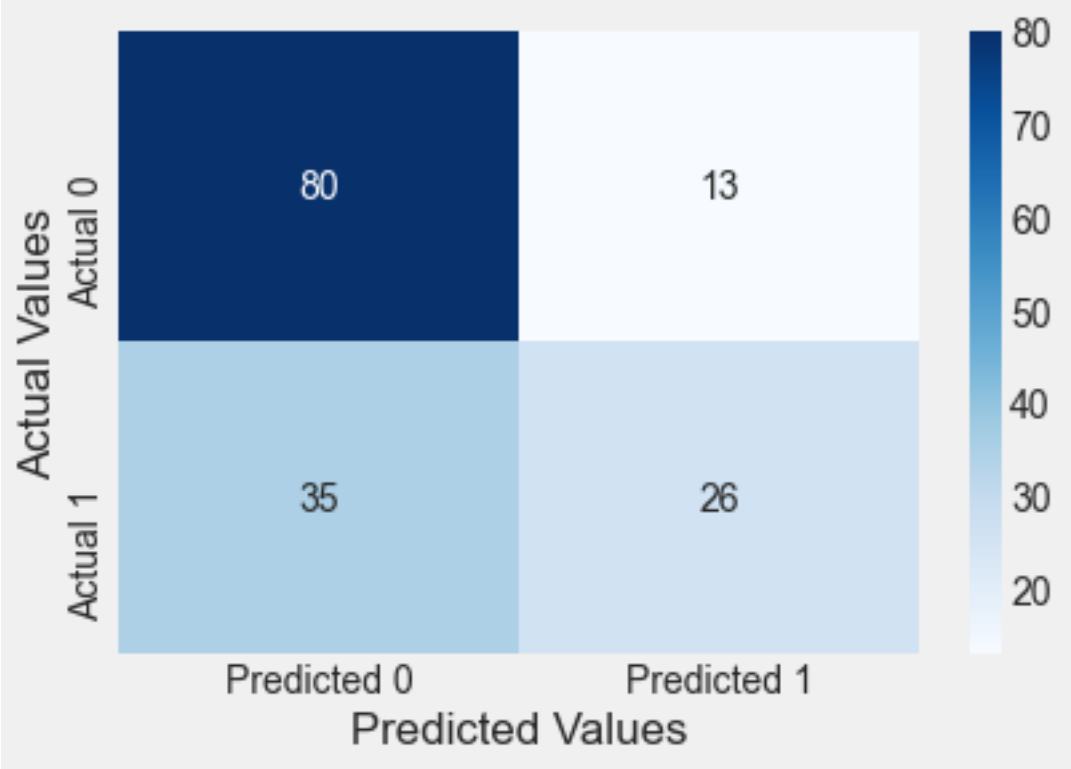
```
confusion_matrix_train(logit_model)
```

```
Classification accuracy = 68.1%
```



```
confusion_matrix_test(test,test.Outcome,logit_model)
```

```
Classification accuracy = 68.8%
```

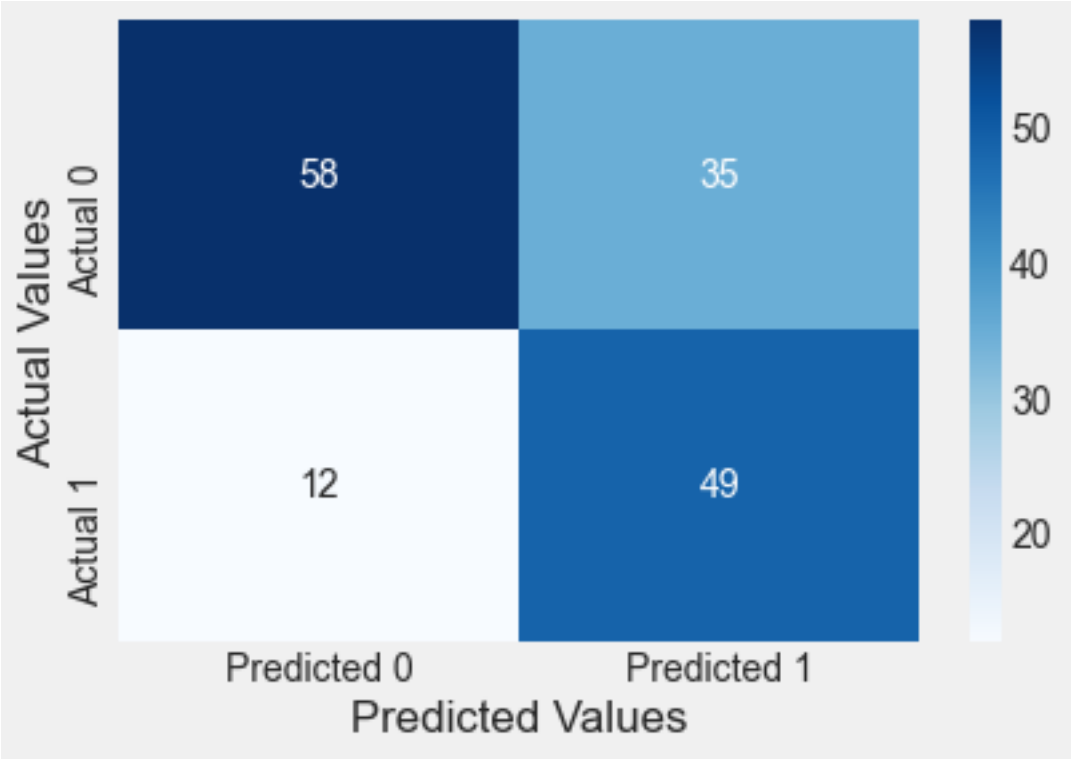


The classification accuracy on test data has increased to 69%. However, the number of *false positives* have increased. But in case of diabetes, *false negatives* are more concerning than *false positives*. This is because if a person has diabetes, and is told that they do not have diabetes, their condition may deteriorate. If a person does not have diabetes, and is told that they have diabetes, they may take unnecessary precautions or tests, but it will not be as harmful to the person as in the previous case. So, in this problem, we will be more focused on reducing the number of *false negatives*, instead of reducing the *false positives* or increasing the overall classification accuracy.

We can decrease the cutoff for classifying a person as having diabetes to reduce the number of false negatives.

```
#Reducing the cutoff for classifying a person as diabetic to 0.3 (instead of 0.5)
confusion_matrix_test(test,test.Outcome,logit_model,0.3)
```

```
Classification accuracy = 69.5%
```



Note that the changed cut-off reduced the number of *false negatives*, but at the cost of increasing the *false positives*. However, the stakeholders may prefer the reduced cut-off to be safer.

Is there another way to transform `Age`?

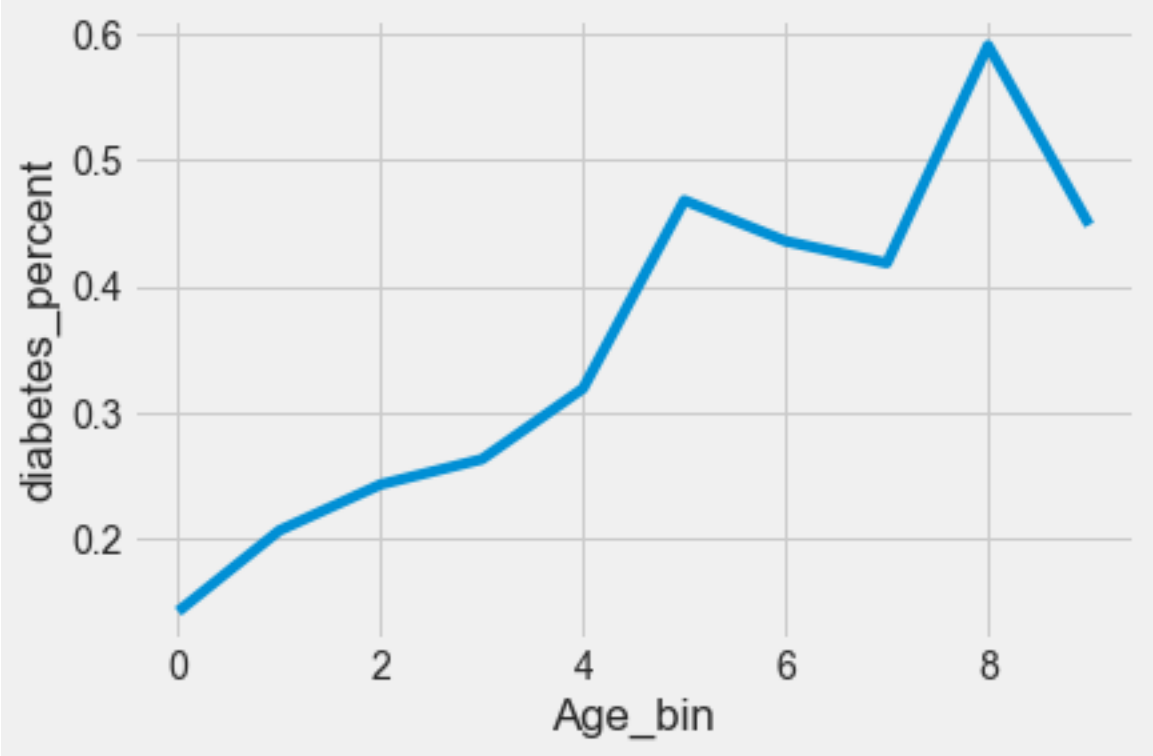
Yes, binning age into bins that have similar proportion of people with diabetes may provide a better model fit.

```
#Creating a function to bin age so that it can be applied to both the test and train datas
def var_transform(data):
    binned_age = pd.qcut(train['Age'],10,retbins=True)
    bins = binned_age[1]
    data['age_binned'] = pd.cut(data['Age'],bins = bins)
    dum = pd.get_dummies(data.age_binned,drop_first = True)
    dum.columns = ['age'+str(x) for x in range(1,len(bins)-1)]
    data = pd.concat([data,dum], axis = 1)
    return data
```

```
#Binning age using the function var_transform()
train = var_transform(train)
test = var_transform(test)
```

```
#Re-creating the plot of diabetes_percent vs age created earlier, just to check if the fun
age_data = train.groupby('age_binned')['Outcome'].agg([('diabetes_percent','mean'),('nobs'
sns.lineplot(x = age_data.index, y= age_data['diabetes_percent'])
plt.xlabel('Age_bin')
```

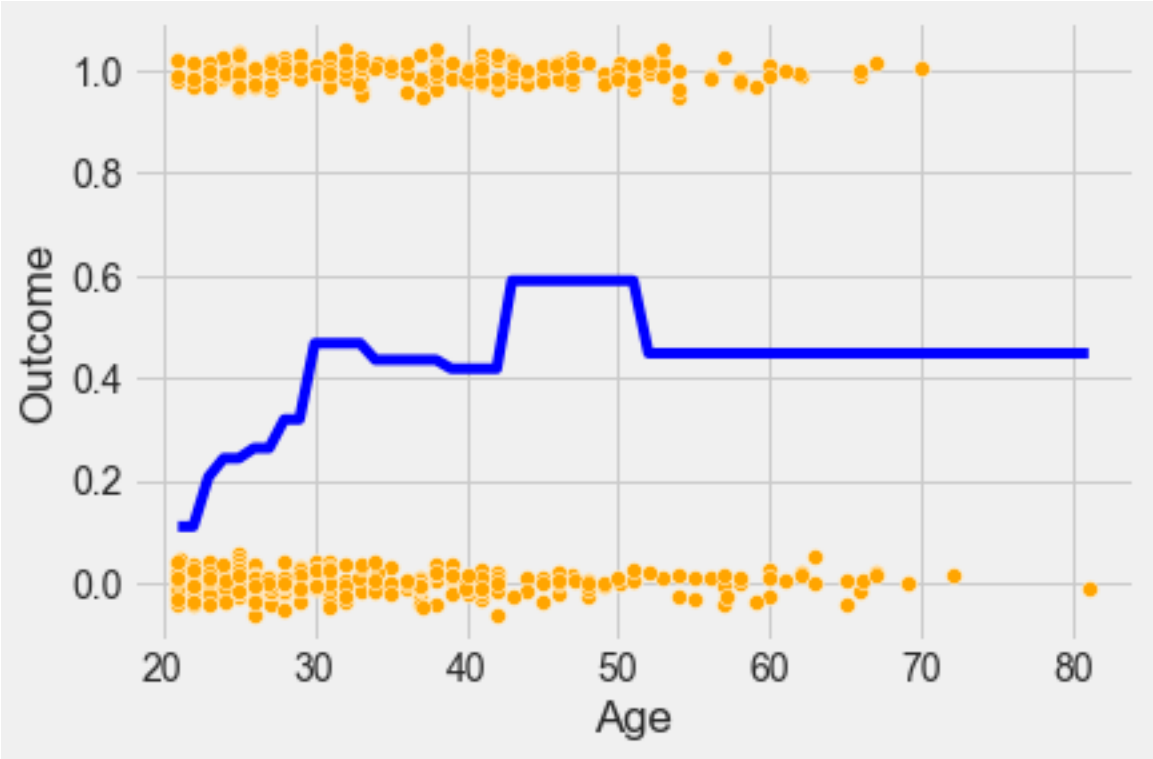
```
Text(0.5, 0, 'Age_bin')
```




```
#Model with binned Age
def jitter(values,j):
    return values + np.random.normal(j,0.02,values.shape)
sns.scatterplot(x = jitter(train.Age,0), y = jitter(train.Outcome,0), data = train, color
logit_model = sm.logit(formula = 'Outcome~' + '+'.join(['age'+str(x) for x in range(1,10)]
sns.lineplot(x = 'Age', y= logit_model.predict(train), data = train, color = 'blue')
```

```
Optimization terminated successfully.
         Current function value: 0.585956
         Iterations 6
```

```
<AxesSubplot:xlabel='Age', ylabel='Outcome'>
```




```
logit_model.summary()
```

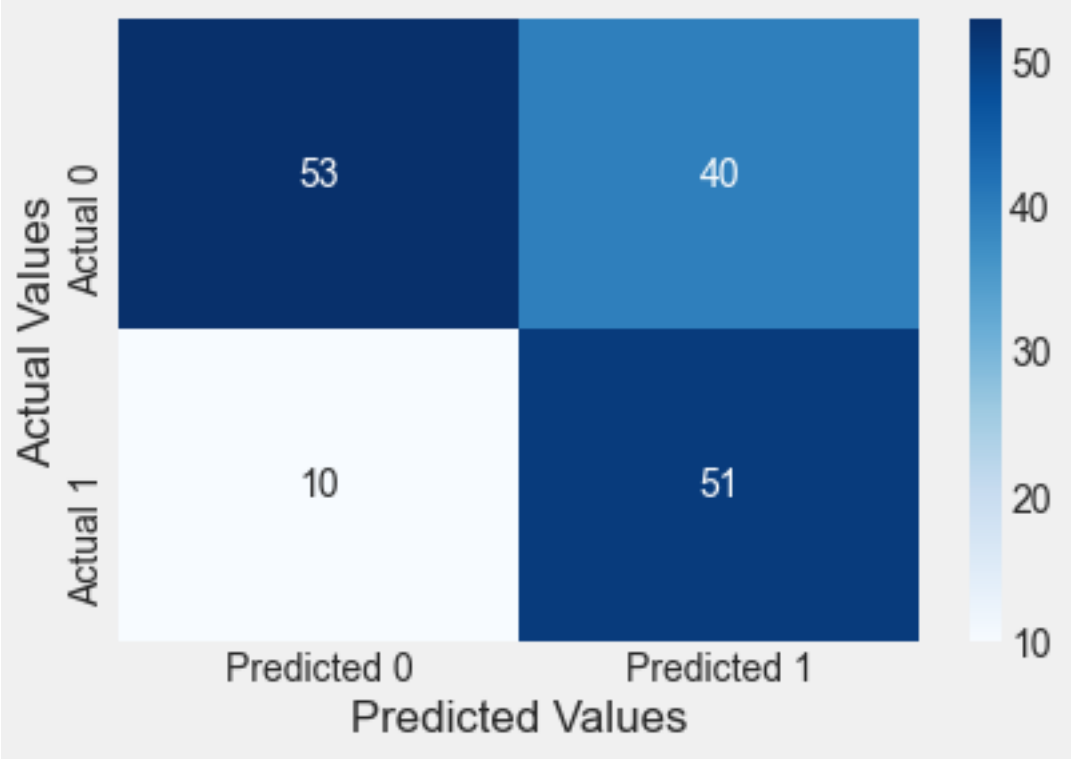
Table 8.9: Logit Regression Results

Dep. Variable:	Outcome	No. Observations:	614
Model:	Logit	Df Residuals:	604
Method:	MLE	Df Model:	9
Date:	Sun, 19 Feb 2023	Pseudo R-squ.:	0.08318
Time:	14:19:51	Log-Likelihood:	-359.78
converged:	True	LL-Null:	-392.42
Covariance Type:	nonrobust	LLR p-value:	1.273e-10

Note that the probability of having diabetes for each age bin is a constant, as per the above plot.

```
confusion_matrix_test(test,test.Outcome,logit_model,0.3)
```

```
Classification accuracy = 67.5%
```



Binning `Age` provides a similar result as compared to the model with the quadratic transformation of `Age`.

```
train.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
158	2	88	74	19	53	29.0	0.229	22
251	2	129	84	0	0	28.0	0.284	27
631	0	102	78	40	90	34.5	0.238	24
757	0	123	72	0	0	36.3	0.258	52

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
689	1	144	82	46	180	46.1	0.335	46

```
#Model with the quadratic transformation of Age and more predictors
logit_model_diabetes = sm.logit(formula = 'Outcome~Age+I(Age**2)+Glucose+BloodPressure+BMI
logit_model_diabetes.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.470478
         Iterations 6
```

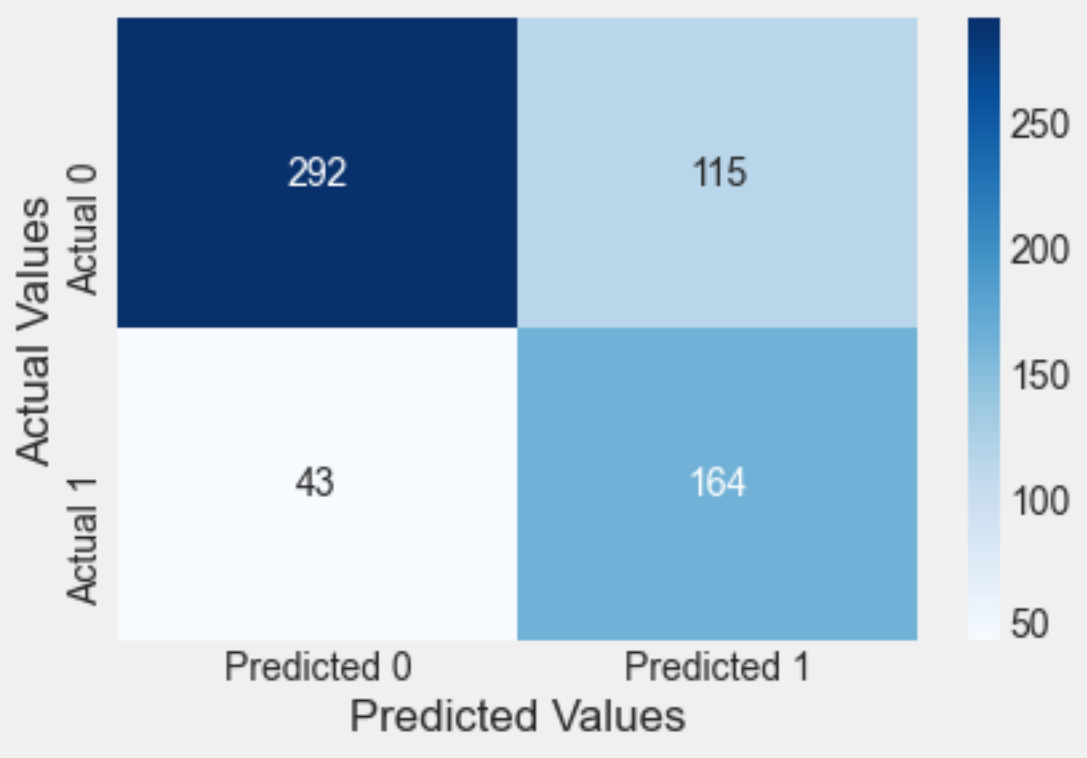
Table 8.11: Logit Regression Results

Dep. Variable:	Outcome	No. Observations:	614
Model:	Logit	Df Residuals:	607
Method:	MLE	Df Model:	6
Date:	Thu, 23 Feb 2023	Pseudo R-squ.:	0.2639
Time:	10:26:00	Log-Likelihood:	-288.87
converged:	True	LL-Null:	-392.42
Covariance Type:	nonrobust	LLR p-value:	5.878e-42

Adding more predictors has increased the log likelihood of the model as expected.

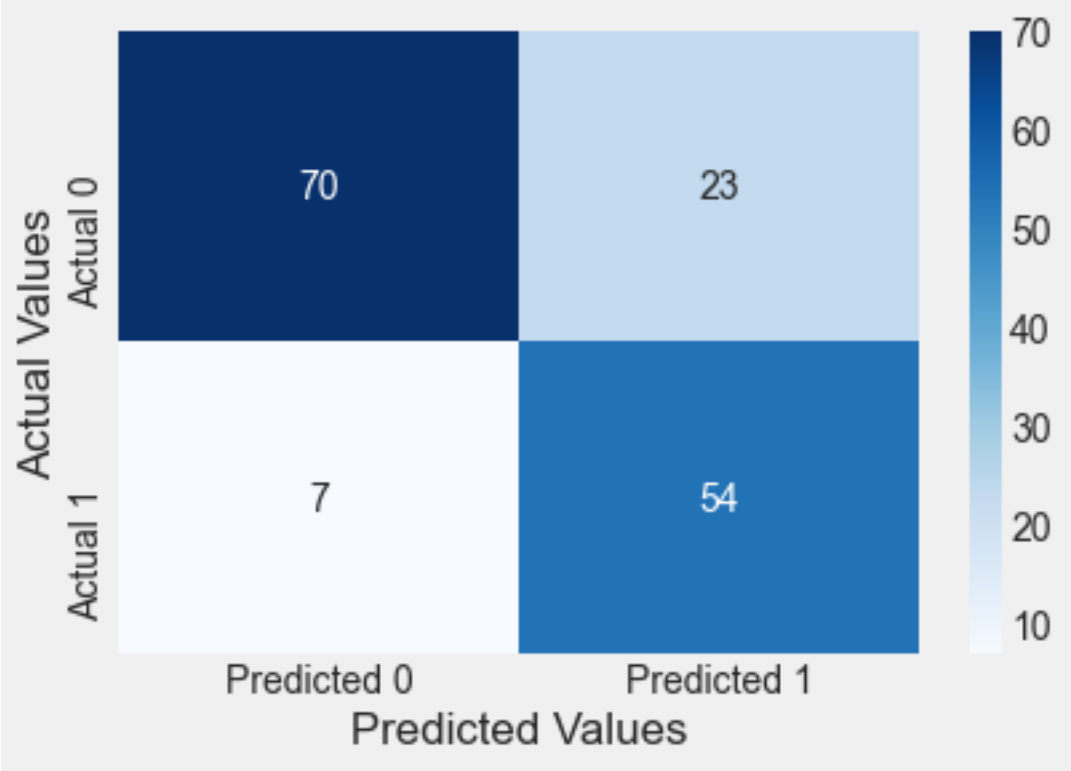
```
confusion_matrix_train(logit_model_diabetes,cutoff=0.3)
```

```
Classification accuracy = 74.3%
```




```
confusion_matrix_test(test,test.Outcome,logit_model_diabetes,0.3)
```

```
Classification accuracy = 80.5%
```



The model with more predictors also has lesser number of *false negatives*, and higher overall classification accuracy.

How many bins must you make for `Age` to get the most accurate model?
If the number of bins are too less, the trend may not be captured accurately. If the number of bins are too many, it may lead to overfitting of the model. There is an optimal value of the number of bins that captures the trend, but does not overfit. A couple of ways of estimating the optimal number of bins can be:

1. The number of bins for which the trend continues to be "almost" the same for several samples of the data.
2. Testing the model on multiple test datasets.

Optimizing the number of bins for each predictor may be a time-consuming exercises. You may do it for your course project. However, we will not do it here in the class notes.

8.5 Performance Measurement

We have already seen the confusion matrix, and classification accuracy. Now, let us see some other useful performance metrics that can be computed from the confusion matrix. The metrics below are computed for the confusion matrix immediately above this section *(or the confusion matrix on test data corresponding to the model* ***`logit_model_diabetes`****).*

8.5.1 Precision-recall

Precision measures the accuracy of positive predictions. Also called the `precision` of the classifier

$$\text{precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

==> `70.13%`

`Precision` is typically used with `recall` (`Sensitivity` or `True Positive Rate`). The ratio of positive instances that are correctly detected by the classifier.

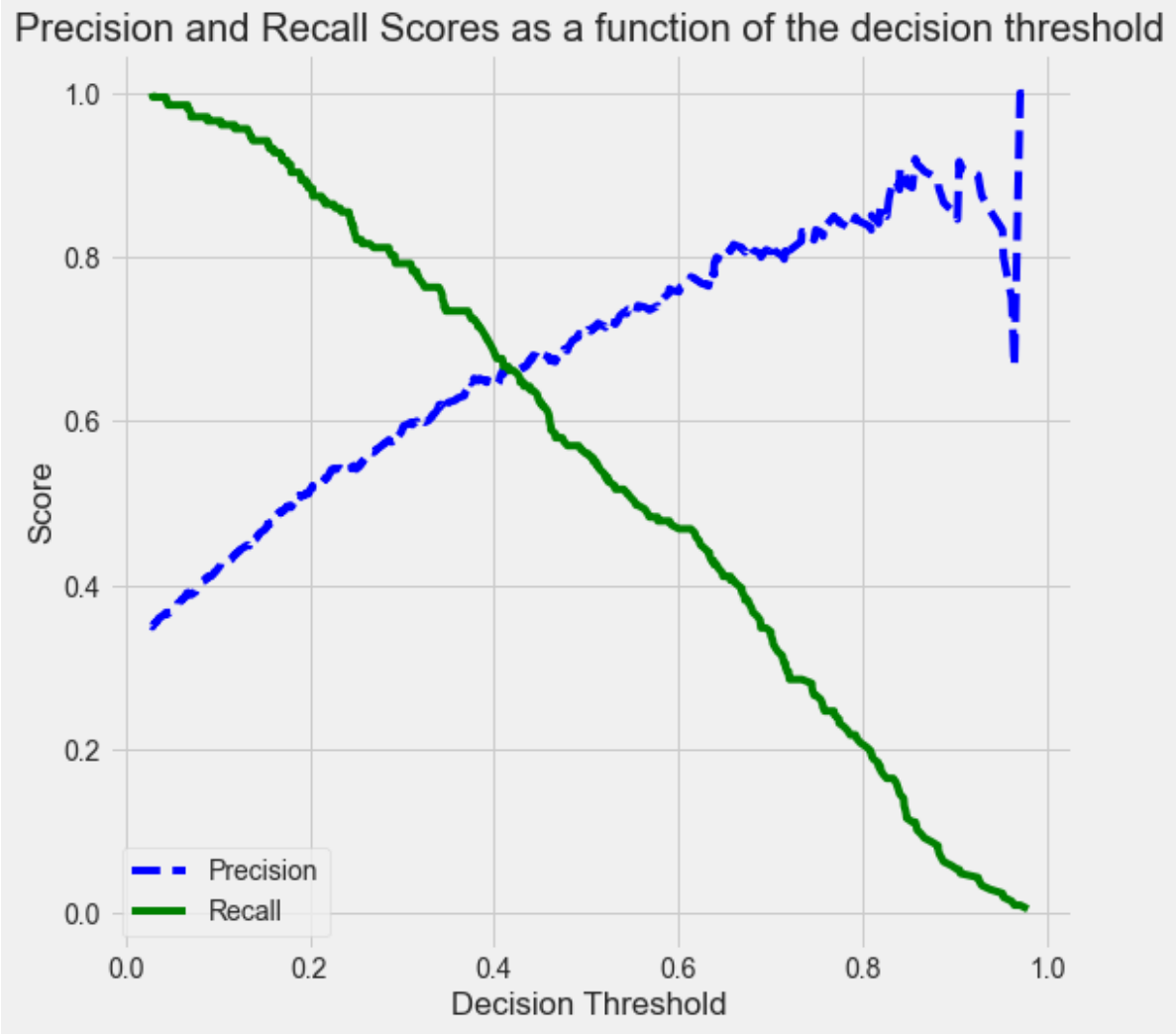
$$\text{recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

==> `88.52%`

Precision / Recall Tradeoff: Increasing precision reduces recall and vice versa.

Visualize the precision-recall curve for the model `logit_model_diabetes`.

```
from sklearn.metrics import precision_recall_curve
y=train.Outcome
ypred = logit_model_diabetes.predict(train)
p, r, thresholds = precision_recall_curve(y, ypred)
def plot_precision_recall_vs_threshold(precisions, recalls, thresholds):
    plt.figure(figsize=(8, 8))
    plt.title("Precision and Recall Scores as a function of the decision threshold")
    plt.plot(thresholds, precisions[:-1], "b--", label="Precision")
    plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
    plt.ylabel("Score")
    plt.xlabel("Decision Threshold")
    plt.legend(loc='best')
    plt.legend()
plot_precision_recall_vs_threshold(p, r, thresholds)
```



As the decision threshold probability increases, the precision increases, while the recall decreases.

Q: How are the values of the `thresholds` chosen to make the precision-recall curve?

Hint: Look at the documentation for `precision_recall_curve`.

8.5.2 The Receiver Operating Characteristics (ROC) Curve

A **ROC(Receiver Operator Characteristic Curve)** is a plot of sensitivity (True Positive Rate) on the y axis against (1−specificity) (False Positive Rate) on the x axis for varying values of the threshold t. The 45° diagonal line connecting (0,0) to (1,1) is the ROC curve

corresponding to random chance. The ROC curve for the gold standard is the line connecting (0,0) to (0,1) and (0,1) to (1,1).

```
<IPython.core.display.Image object>
```

```
<IPython.core.display.Image object>
```

An animation to demonstrate how an ROC curve relates to sensitivity and specificity for all possible cutoffs (Source)

High Threshold:

- High specificity
- Low sensitivity

Low Threshold

- Low specificity
- High sensitivity

The area under ROC is called *Area Under the Curve(AUC)*. AUC gives the rate of successful classification by the logistic model. To get a more in-depth idea of what a ROC-AUC curve is and how is it calculated, here is a good blog link.

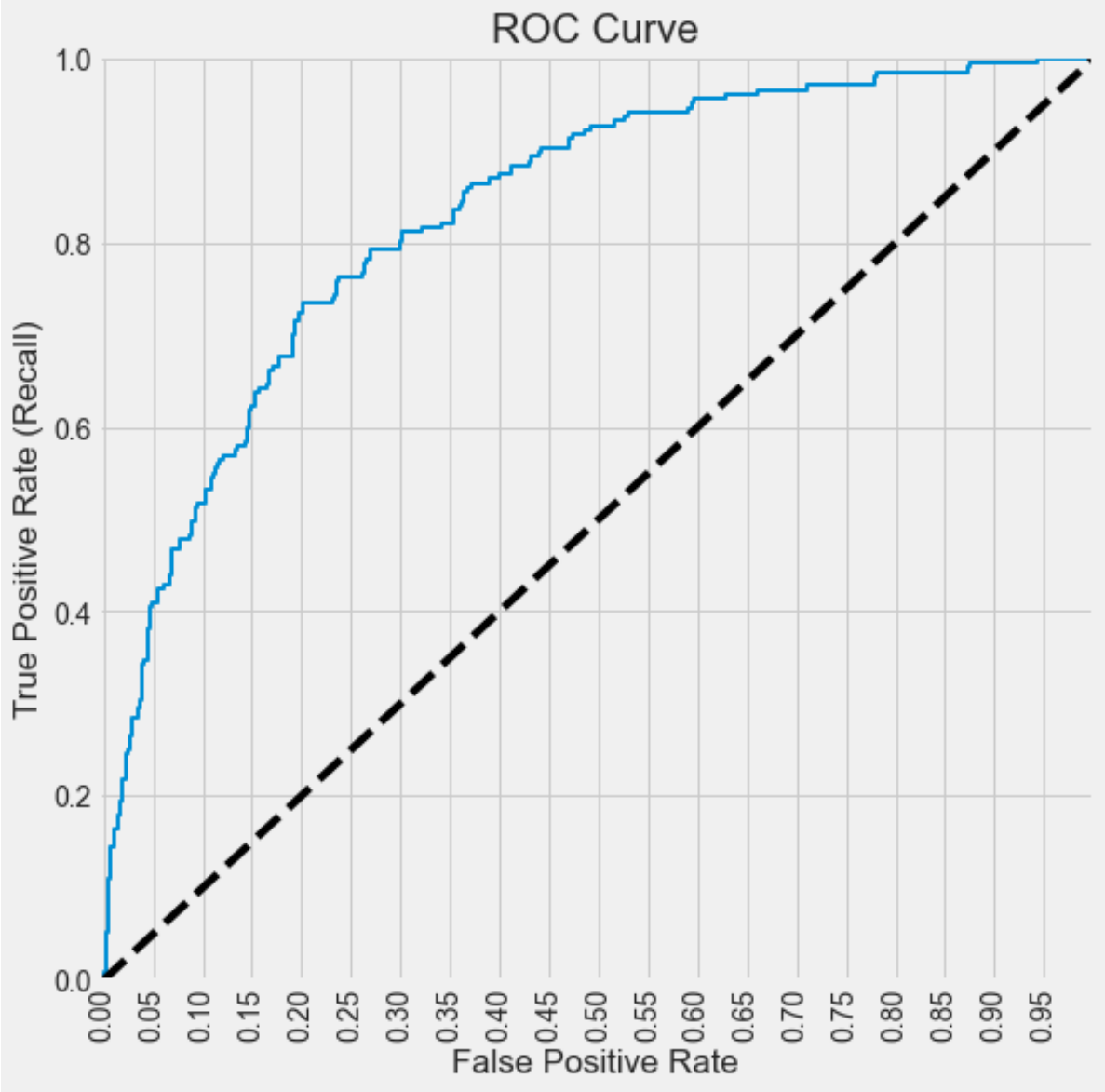
Here is good post by google developers on interpreting ROC-AUC, and its advantages / disadvantages.

Visualize the ROC curve and compute the ROC-AUC for the model `logit_model_diabetes`.

```
from sklearn.metrics import roc_curve, auc
y=train.Outcome
ypred = logit_model_diabetes.predict(train)
fpr, tpr, auc_thresholds = roc_curve(y, ypred)
print(auc(fpr, tpr))# AUC of ROC
def plot_roc_curve(fpr, tpr, label=None):
    plt.figure(figsize=(8,8))
    plt.title('ROC Curve')
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.axis([-0.005, 1, 0, 1.005])
    plt.xticks(np.arange(0,1, 0.05), rotation=90)
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate (Recall)")
```

```
fpr, tpr, auc_thresholds = roc_curve(y, ypred)
plot_roc_curve(fpr, tpr)
```

```
0.8325914847653979
```



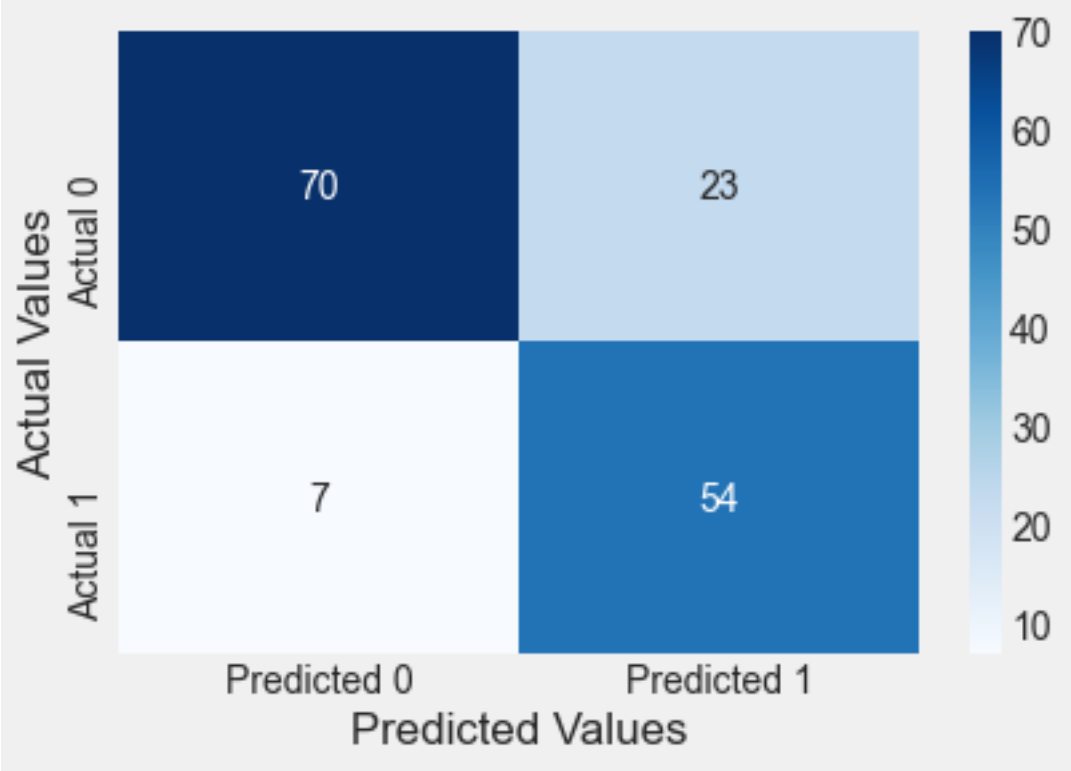
Q: How are the values of the `auc_thresholds` chosen to make the ROC curve? Why does it look like a step function?

Below is a function that prints the confusion matrix along with all the performance metrics we discussed above for a given decision threshold probability, on train / test data. Note that ROC-AUC does not depend on a decision threshold probability.

```
#Function to compute confusion matrix and prediction accuracy on test/train data
def confusion_matrix_data(data,actual_values,model,cutoff=0.5):
#Predict the values using the Logit model
    pred_values = model.predict(data)
# Specify the bins
    bins=np.array([0,cutoff,1])
#Confusion matrix
    cm = np.histogram2d(actual_values, pred_values, bins=bins)[0]
    cm_df = pd.DataFrame(cm)
    cm_df.columns = ['Predicted 0','Predicted 1']
    cm_df = cm_df.rename(index={0: 'Actual 0',1:'Actual 1'})
# Calculate the accuracy
    accuracy = (cm[0,0]+cm[1,1])/cm.sum()
    fnr = (cm[1,0])/(cm[1,0]+cm[1,1])
    precision = (cm[1,1])/(cm[0,1]+cm[1,1])
    fpr = (cm[0,1])/(cm[0,0]+cm[0,1])
    tpr = (cm[1,1])/(cm[1,0]+cm[1,1])
    fpr_roc, tpr_roc, auc_thresholds = roc_curve(actual_values, pred_values)
    auc_value = (auc(fpr_roc, tpr_roc))# AUC of ROC
    sns.heatmap(cm_df, annot=True, cmap='Blues', fmt='g')
    plt.ylabel("Actual Values")
    plt.xlabel("Predicted Values")
    print("Classification accuracy = {:.1%}".format(accuracy))
    print("Precision = {:.1%}".format(precision))
    print("TPR or Recall = {:.1%}".format(tpr))
    print("FNR = {:.1%}".format(fnr))
    print("FPR = {:.1%}".format(fpr))
    print("ROC-AUC = {:.1%}".format(auc_value))
```

```
confusion_matrix_data(test,test.Outcome,logit_model_diabetes,0.3)
```

```
Classification accuracy = 80.5%
Precision = 70.1%
TPR or Recall = 88.5%
FNR = 11.5%
FPR = 24.7%
ROC-AUC = 90.1%
```


70
23
7
54
Actual Values
Actual 0
Actual 1
Predicted 0
Predicted 1
Predicted Values
70
60
50
40
30
20
10

Part III

Variable selection & Regularization

9 Best subset and Stepwise selection

Read section 6.1 of the book before using these notes.

Note that in this course, lecture notes are not sufficient, you must read the book for better understanding. Lecture notes are just implementing the concepts of the book on a dataset, but not explaining the concepts elaborately.

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as sm
import seaborn as sns
import matplotlib.pyplot as plt
import itertools
import time
```

```
trainf = pd.read_csv('./Datasets/house_feature_train.csv')
trainp = pd.read_csv('./Datasets/house_price_train.csv')
testf = pd.read_csv('./Datasets/house_feature_test.csv')
testp = pd.read_csv('./Datasets/house_price_test.csv')
train = pd.merge(trainf,trainp)
test = pd.merge(testf,testp)
train.head()
```

	house_id	house_age	distance_MRT	number_convenience_stores	latitude	longitude	house_pri
0	210	5.2	390.5684	5	24.97937	121.54245	2724.84
1	190	35.3	616.5735	8	24.97945	121.53642	1789.29
2	328	15.9	1497.7130	3	24.97003	121.51696	556.96
3	5	7.1	2175.0300	3	24.96305	121.51254	1030.41
4	412	8.1	104.8101	5	24.96674	121.54067	2756.25

Develop a model to predict house price using the rest of the columns as predictors (except `house_id`).

```
#Model with log house price as the response and the remaining variables as predictors
model = sm.ols('np.log(house_price)~house_age+distance_MRT+number_convenience_stores+latit
longitude', data = train).fit()
model.summary()
```

Table 9.2: OLS Regression Results

Dep. Variable:	np.log(house_price)	R-squared:	0.772
Model:	OLS	Adj. R-squared:	0.767
Method:	Least Squares	F-statistic:	181.8
Date:	Thu, 16 Feb 2023	Prob (F-statistic):	4.47e-84
Time:	18:31:07	Log-Likelihood:	-118.47
No. Observations:	275	AIC:	248.9
Df Residuals:	269	BIC:	270.6
Df Model:	5		
Covariance Type:	nonrobust		

Find the best subset of predictors that can predict house price in a linear regression model.

```
#Creating a set of predictors from which we need to find the best subset of predictors
X = train[['house_age','number_convenience_stores','latitude', 'longitude','distance_MRT']
```

9.0.1 Best subset selection algorithm

Now, we will implement the algorithm of finding the best subset of predictors from amongst all sets of predictors.

```
#Function to develop a model based on all predictors in predictor_subset
def processSubset(predictor_subset):
    # Fit model on feature_set and calculate R-squared
    model = sm.ols('np.log(house_price)~' + '+'.join(predictor_subset),data = train).fit()
    Rsquared = model.rsquared
    return {"model":model, "Rsquared":Rsquared}
```

```
#Function to select the best model amongst all models with 'k' predictors
def getBest_model(k):
    tic = time.time()
    results = []
```

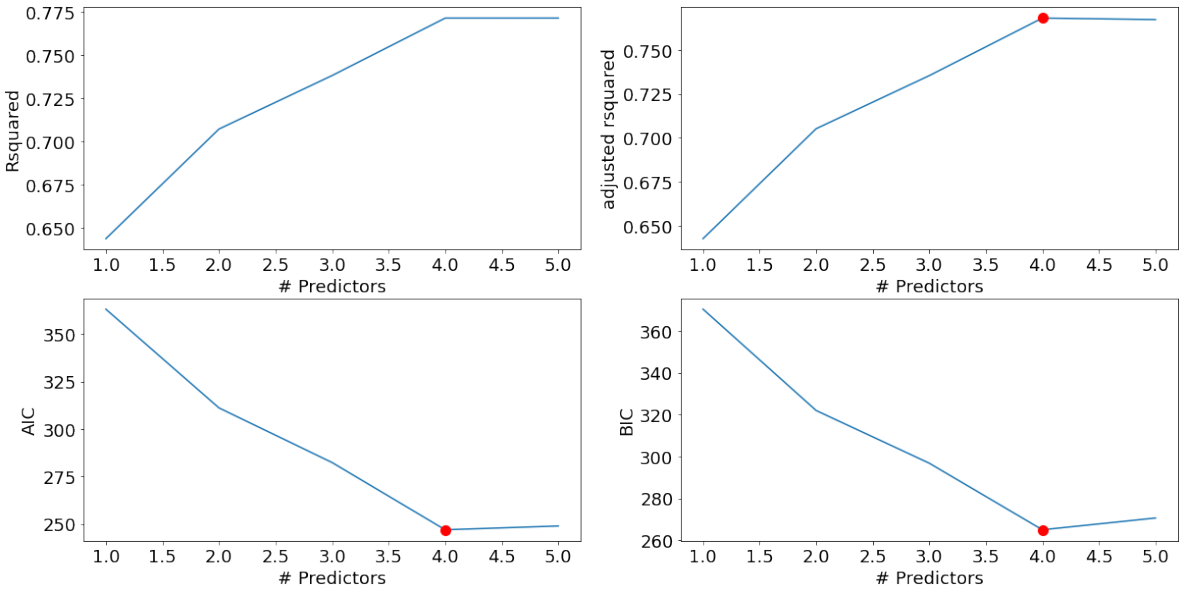
```
    for combo in itertools.combinations(X.columns, k):
        results.append(processSubset((list(combo))))

    # Wrap everything up in a dataframe
    models = pd.DataFrame(results)

    # Choose the model with the highest RSS
    best_model = models.loc[models['Rsquared'].argmax()]

    toc = time.time()
    print("Processed", models.shape[0], "models on", k, "predictors in", (toc-tic), "secon
    return best_model
```

```
#Function to select the best model amongst the best models for 'k' predictors, where k = 1
models_best = pd.DataFrame(columns=["Rsquared", "model"])

tic = time.time()
for i in range(1,1+X.shape[1]):
    models_best.loc[i] = getBest_model(i)

toc = time.time()
print("Total elapsed time:", (toc-tic), "seconds.")
```

```
Processed 5 models on 1 predictors in 0.02393651008605957 seconds.
Processed 10 models on 2 predictors in 0.04688239097595215 seconds.
Processed 10 models on 3 predictors in 0.04986691474914551 seconds.
Processed 5 models on 4 predictors in 0.029920578002929688 seconds.
Processed 1 models on 5 predictors in 0.008975982666015625 seconds.
Total elapsed time: 0.17253828048706055 seconds.
```

```
def best_sub_plots():
    plt.figure(figsize=(20,10))
    plt.rcParams.update({'font.size': 18, 'lines.markersize': 10})

    # Set up a 2x2 grid so we can look at 4 plots at once
    plt.subplot(2, 2, 1)

    # We will now plot a red dot to indicate the model with the largest adjusted R^2 stati
    # The argmax() function can be used to identify the location of the maximum point of a
    plt.plot(models_best["Rsquared"])
```

```
    plt.xlabel('# Predictors')
    plt.ylabel('Rsquared')

    # We will now plot a red dot to indicate the model with the largest adjusted R^2 stati
    # The argmax() function can be used to identify the location of the maximum point of a

    rsquared_adj = models_best.apply(lambda row: row[1].rsquared_adj, axis=1)

    plt.subplot(2, 2, 2)
    plt.plot(rsquared_adj)
    plt.plot(1+rsquared_adj.argmax(), rsquared_adj.max(), "or")
    plt.xlabel('# Predictors')
    plt.ylabel('adjusted rsquared')

    # We'll do the same for AIC and BIC, this time looking for the models with the SMALLES
    aic = models_best.apply(lambda row: row[1].aic, axis=1)

    plt.subplot(2, 2, 3)
    plt.plot(aic)
    plt.plot(1+aic.argmin(), aic.min(), "or")
    plt.xlabel('# Predictors')
    plt.ylabel('AIC')

    bic = models_best.apply(lambda row: row[1].bic, axis=1)

    plt.subplot(2, 2, 4)
    plt.plot(bic)
    plt.plot(1+bic.argmin(), bic.min(), "or")
    plt.xlabel('# Predictors')
    plt.ylabel('BIC')
best_sub_plots()
```



The model with 4 predictors is the best model, according to all 3 criteria - Adjusted R-squared, AIC and BIC.

Note that we have not considered the null model (i.e., the model with only the intercept and no predictors) explicitly in the best subsets algorithm. However, the null model is considered when selecting the best model. The R-squared and the adjusted R-squared for the null model is 0. So, if the adjusted R-squared of all the models with at least one predictor is negative, then the null model will be the best model.

```
best_subset_model = models_best.loc[4,'model']
models_best.loc[4,'model'].summary()
```

Table 9.3: OLS Regression Results

Dep. Variable:	np.log(house_price)	R-squared:	0.772
Model:	OLS	Adj. R-squared:	0.768
Method:	Least Squares	F-statistic:	228.0
Date:	Thu, 16 Feb 2023	Prob (F-statistic):	2.79e-85
Time:	19:51:50	Log-Likelihood:	-118.47
No. Observations:	275	AIC:	246.9
Df Residuals:	270	BIC:	265.0
Df Model:	4		
Covariance Type:	nonrobust		

```
#Finding the RMSE of the model selected using the best subset selection procedure
pred_price = np.exp(best_subset_model.predict(test))
np.sqrt(((pred_price - test.house_price)**2).mean())
```

```
403.4635674362065
```

```
#RMSE of the model using all the predictors
model = sm.ols('np.log(house_price)~' + '+'.join(X.columns),data = train).fit()
pred_price = np.exp(model.predict(test))
np.sqrt(((pred_price - test.house_price)**2).mean())
```

```
403.8409399214197
```

The RMSE of the best subset model is similar to the RMSE of the model with all the predictors. This is because longitude varies only in [121.47, 121.57]. The coefficient of longitude is 0.1923 in the model with all the predictors. So, the change in the response due to longitude is in [23.36, 23.38]. This change in the response due to longitude is almost a constant, and hence is adjusted in the intercept of the model without longitude. Note the intercept of the model without longitude is 23.91 more than the intercept of the model with longitude.

```
[0.1923*train.longitude.min(),0.1923*train.longitude.max()]
```

```
[23.359359818999998, 23.377193721]
```

9.0.2 Including interactions for best subset selection

Let's perform best subset selection including all the predictors and their 2-factor interactions

```
#Creating a dataframe with all the predictors
X = train[['house_age', 'distance_MRT', 'number_convenience_stores','latitude','longitude'
#Since 'X' will change when we include interactions, we need a backup containing all indiv
X_backup = train[['house_age', 'distance_MRT', 'number_convenience_stores','latitude','lon
```

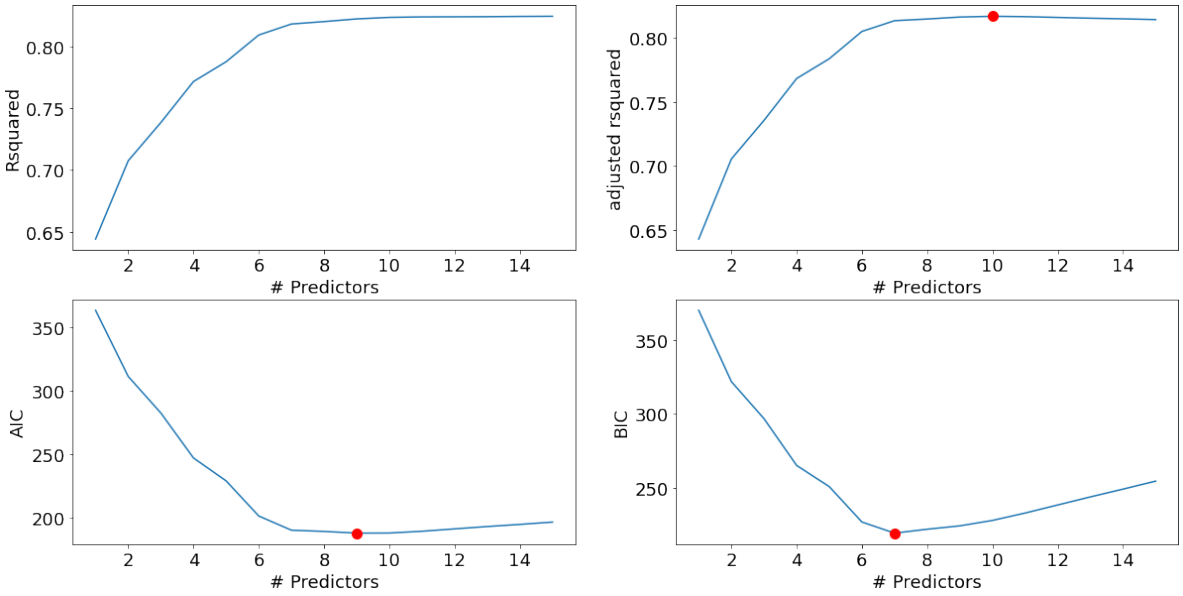
```
#Including 2-factor interactions of predictors in train and 'X'. Note that we need train t
#find 'k' variable subsets from amongst all the predictors under consideration
for combo in itertools.combinations(X_backup.columns, 2):
    train['_'.join(combo)] = train[combo[0]]*train[combo[1]]
```

```
    test['_'.join(combo)] = test[combo[0]]*test[combo[1]]
    X.loc[:,'_'.join(combo)] = train.loc[:,'_'.join(combo)]
```

```
models_best = pd.DataFrame(columns=["Rsquared", "model"])

tic = time.time()
for i in range(1,1+X.shape[1]):
    models_best.loc[i] = getBest_model(i)

toc = time.time()
print("Total elapsed time:", (toc-tic), "seconds.")
```

```
Processed 15 models on 1 predictors in 0.07200050354003906 seconds.
Processed 105 models on 2 predictors in 0.536522388458252 seconds.
Processed 455 models on 3 predictors in 2.6639997959136963 seconds.
Processed 1365 models on 4 predictors in 9.176022052764893 seconds.
Processed 3003 models on 5 predictors in 24.184194803237915 seconds.
Processed 5005 models on 6 predictors in 43.54697918891907 seconds.
Processed 6435 models on 7 predictors in 65.83688187599182 seconds.
Processed 6435 models on 8 predictors in 78.97277760505676 seconds.
Processed 5005 models on 9 predictors in 64.53991365432739 seconds.
Processed 3003 models on 10 predictors in 38.39328980445862 seconds.
Processed 1365 models on 11 predictors in 18.715795755386353 seconds.
Processed 455 models on 12 predictors in 6.93279504776001 seconds.
Processed 105 models on 13 predictors in 1.6240253448486328 seconds.
Processed 15 models on 14 predictors in 0.256000280380249 seconds.
Processed 1 models on 15 predictors in 0.024001121520996094 seconds.
Total elapsed time: 356.2638840675354 seconds.
```

```
best_sub_plots()
```



The model with 7 predictors is the best model based on the BIC criterion, and very close to the best model based on the AIC and Adjusted R-squared criteria. Let us select the model with 7 predictors.

```
best_interaction_model = models_best['model'][7]
best_interaction_model.summary()
```

Table 9.4: OLS Regression Results

Dep. Variable:	np.log(house_price)	R-squared:	0.818
Model:	OLS	Adj. R-squared:	0.814
Method:	Least Squares	F-statistic:	171.7
Date:	Thu, 16 Feb 2023	Prob (F-statistic):	5.29e-95
Time:	20:17:02	Log-Likelihood:	-87.046
No. Observations:	275	AIC:	190.1
Df Residuals:	267	BIC:	219.0
Df Model:	7		
Covariance Type:	nonrobust		

Note that only 3 of the 10 two factor interactions are included in the best subset model, and the predictor `longitude` has been dropped.

```
#Finding the RMSE of the model selected using the best subset selection procedure, where t
#include 2-factor interactions
pred_price = np.exp(best_interaction_model.predict(test))
np.sqrt(((pred_price - test.house_price)**2).mean())
```

```
346.4100962681362
```

```
#Model with the predictors and all their 2-factor interactions
model = sm.ols('np.log(house_price)~' + '+'.join(X.columns),data = train).fit()
model.summary()
```

Table 9.5: OLS Regression Results

Dep. Variable:	np.log(house_price)	R-squared:	0.825
Model:	OLS	Adj. R-squared:	0.814
Method:	Least Squares	F-statistic:	81.14
Date:	Thu, 16 Feb 2023	Prob (F-statistic):	1.33e-88
Time:	20:13:01	Log-Likelihood:	-82.228
No. Observations:	275	AIC:	196.5
Df Residuals:	259	BIC:	254.3
Df Model:	15		
Covariance Type:	nonrobust		

```
# RMSE of the model using all the predictors and their 2-factor interactions
pred_price = np.exp(model.predict(test))
np.sqrt(((pred_price - test.house_price)**2).mean())
```

```
360.40099598821615
```

The best subset model seems to be slightly better than the model with all the predictors, based on the RMSE on test data.

9.1 Stepwise selection

Best subset selection cannot be used in case of even a slightly large number of predictors. In the previous example, we had 15 predictors. The number of models that we developed to find the best subset of predictors from the set of 15 predictors was $2^{15} \approx 32,000$. In case of 20

predictors, the number of models to use the best subset selection approach will be $2^{20} \approx 1$ million, which is computationally too expensive. Due to this limitation of the best subsets selection method, we will use stepwise regression, which explores a far more restricted set of models, and thus is an attractive alternative to the best subset selection method.

9.2 Forward stepwise selection

Source - Page 229: "Forward stepwise selection is a computationally efficient alternative to best subset selection. While the best subset selection procedure considers all 2^p possible models containing subsets of the p predictors, forward stepwise considers a much smaller set of models. Forward stepwise selection begins with a model containing no predictors, and then adds predictors to the model, one-at-a-time, until all of the predictors are in the model. In particular, at each step the variable that gives the greatest additional improvement to the fit is added to the model."

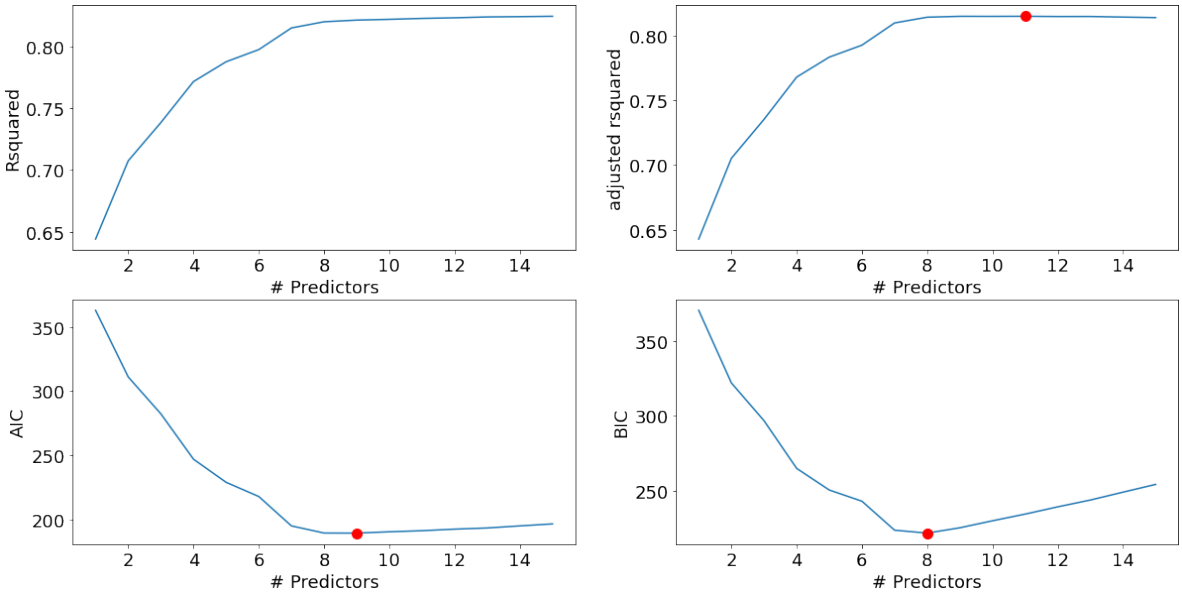
```
#Function to find the best predictor out of p-k predictors and add it to the model contain
def forward(predictors):

    # Pull out predictors we still need to process
    remaining_predictors = [p for p in X.columns if p not in predictors]

    tic = time.time()

    results = []

    for p in remaining_predictors:
        results.append(processSubset(predictors+[p]))

    # Wrap everything up in a nice dataframe
    models = pd.DataFrame(results)

    # Choose the model with the highest RSS
    best_model = models.loc[models['Rsquared'].argmax()]

    toc = time.time()
    print("Processed ", models.shape[0], "models on", len(predictors)+1, "predictors in",

    # Return the best model, along with some other useful information about the model
    return best_model
```

```
def forward_selection():
    models_best = pd.DataFrame(columns=["Rsquared", "model"])

    tic = time.time()
    predictors = []

    for i in range(1,len(X.columns)+1):
        models_best.loc[i] = forward(predictors)
        predictors = list(models_best.loc[i]["model"].params.index[1:])

    toc = time.time()
    print("Total elapsed time:", (toc-tic), "seconds.")
    return models_best
```

```
models_best = forward_selection()
```

```
Processed  15 models on 1 predictors in 0.06280803680419922 seconds.
Processed  14 models on 2 predictors in 0.054885149002075195 seconds.
Processed  13 models on 3 predictors in 0.05983686447143555 seconds.
Processed  12 models on 4 predictors in 0.06781768798828125 seconds.
Processed  11 models on 5 predictors in 0.07380270957946777 seconds.
Processed  10 models on 6 predictors in 0.07380390167236328 seconds.
Processed  9 models on 7 predictors in 0.06981182098388672 seconds.
Processed  8 models on 8 predictors in 0.07480072975158691 seconds.
Processed  7 models on 9 predictors in 0.0718071460723877 seconds.
Processed  6 models on 10 predictors in 0.06380081176757812 seconds.
Processed  5 models on 11 predictors in 0.054854631423950195 seconds.
Processed  4 models on 12 predictors in 0.05385565757751465 seconds.
Processed  3 models on 13 predictors in 0.04188799858093262 seconds.
Processed  2 models on 14 predictors in 0.027925491333007812 seconds.
Processed  1 models on 15 predictors in 0.016956090927124023 seconds.
Total elapsed time: 0.9055600166320801 seconds.
```

```
best_sub_plots()
```



The model with 8 predictors is the best model based on the BIC criterion, and very close to the best model based on the AIC and Adjusted R-squared criteria. Let us select the model with 8 predictors.

```
best_fwd_reg_model = models_best['model'][8]
best_fwd_reg_model.summary()
```

Table 9.6: OLS Regression Results

Dep. Variable:	np.log(house_price)	R-squared:	0.820
Model:	OLS	Adj. R-squared:	0.815
Method:	Least Squares	F-statistic:	151.6
Date:	Thu, 16 Feb 2023	Prob (F-statistic):	1.91e-94
Time:	20:35:14	Log-Likelihood:	-85.667
No. Observations:	275	AIC:	189.3
Df Residuals:	266	BIC:	221.9
Df Model:	8		
Covariance Type:	nonrobust		

```
#Finding the RMSE of the model selected using the forward selection procedure, where the p
#include 2-factor interactions
pred_price = np.exp(best_fwd_reg_model.predict(test))
np.sqrt(((pred_price - test.house_price)**2).mean())
```

```
364.2004089481364
```

We get a different model than what we got with the best subsets selection method. However, we got it in 0.9 seconds, instead of 6 minutes taken by the best subset selection algorithm. Note that this model has a higher RMSE as compared to the model obtained with the best subset selection procedure, which is expected. However, the RMSE is even slightly higher than the model that includes all the two factor interactions. This may be due to the following reasons:

- This may be due to chance - the test data set may be biased.
- The stepwise variable selection algorithms are greedy algorithms, and certainly don't guarantee the best model, or even a model better than the one without variable selection. However, in general, they are likely to provide a better model than the base model that includes all the predictors, especially if there are several predictors that are not associated with the response.
- For metrics such as adjusted R-squared, the adjustment is not directly tied to the model being more accurate on test data. The adjustment only ensures that the adjusted R-squared increases if the added predictor sufficiently reduces the RSS (Residual sum of squares) on training data.
- AIC is an unbiased estimate of test error. However, AIC will have some variance as we are using sample data for training the model.

9.3 Backward Stepwise Selection

Source - Page 231: "Like forward stepwise selection, backward stepwise selection provides an efficient alternative to best subset selection. However, unlike forward stepwise selection, it begins with the full least squares model containing all p predictors, and then iteratively removes the least useful predictor, one-at-a-time."

Let us try the backward selection procedure on the model with 15 predictors - *house_age, distance_MRT, number_convenience_stores, latitude, longitude* and their 2-factor interactions.

```
def backward(predictors):

    tic = time.time()

    results = []

    for combo in itertools.combinations(predictors, len(predictors)-1):
        results.append(processSubset(combo))
```

```
    # Wrap everything up in a nice dataframe
    models = pd.DataFrame(results)

    # Choose the model with the highest RSS
    best_model = models.loc[models['Rsquared'].argmax()]

    toc = time.time()
    print("Processed ", models.shape[0], "models on", len(predictors)-1, "predictors in",

    # Return the best model, along with some other useful information about the model
    return best_model
```

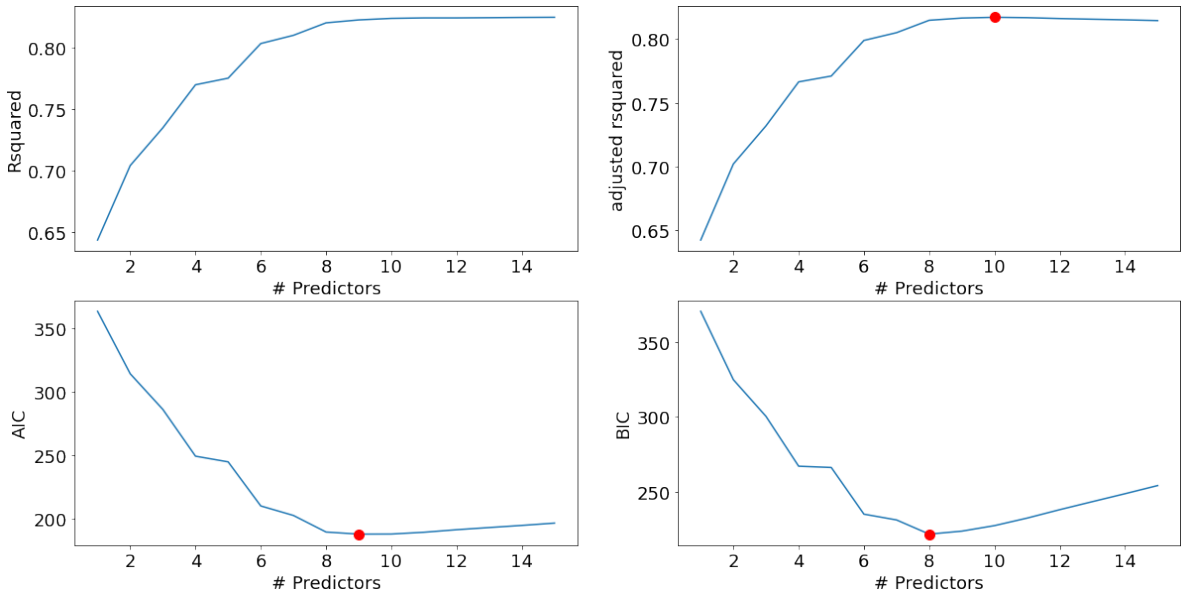
```
def backward_selection():
    models_best = pd.DataFrame(columns=["Rsquared", "model"], index = range(1,len(X.column

    tic = time.time()
    predictors = X.columns
    models_best.loc[len(predictors)] = processSubset(predictors)

    while(len(predictors) > 1):
        models_best.loc[len(predictors)-1] = backward(predictors)
        predictors = models_best.loc[len(predictors)-1]["model"].params.index[1:]

    toc = time.time()
    print("Total elapsed time:", (toc-tic), "seconds.")
    return models_best
```

```
models_best = backward_selection()
```

```
Processed  15 models on 14 predictors in 0.24733757972717285 seconds.
Processed  14 models on 13 predictors in 0.1765275001525879 seconds.
Processed  13 models on 12 predictors in 0.16356277465820312 seconds.
Processed  12 models on 11 predictors in 0.13364267349243164 seconds.
Processed  11 models on 10 predictors in 0.11968183517456055 seconds.
Processed  10 models on 9 predictors in 0.09571337699890137 seconds.
Processed  9 models on 8 predictors in 0.08377647399902344 seconds.
Processed  8 models on 7 predictors in 0.06981253623962402 seconds.
Processed  7 models on 6 predictors in 0.048902273178100586 seconds.
Processed  6 models on 5 predictors in 0.04088902473449707 seconds.
Processed  5 models on 4 predictors in 0.029920101165771484 seconds.
```

```
Processed  4 models on 3 predictors in 0.020944595336914062 seconds.
Processed  3 models on 2 predictors in 0.013962507247924805 seconds.
Processed  2 models on 1 predictors in 0.007978677749633789 seconds.
Total elapsed time: 1.286529779434204 seconds.
```

```
best_sub_plots()
```




```
best_bwd_reg_model = models_best['model'][8]
best_bwd_reg_model.summary()
```

Table 9.7: OLS Regression Results

Dep. Variable:	np.log(house_price)	R-squared:	0.820
Model:	OLS	Adj. R-squared:	0.815
Method:	Least Squares	F-statistic:	151.5
Date:	Thu, 16 Feb 2023	Prob (F-statistic):	2.00e-94
Time:	20:40:43	Log-Likelihood:	-85.714
No. Observations:	275	AIC:	189.4
Df Residuals:	266	BIC:	222.0
Df Model:	8		
Covariance Type:	nonrobust		

We get a slightly different model than what we got with the best subsets selection method

and the forward selection method. As in forward selection, we got it relatively very quickly (in 1.28 seconds), instead of 6 minutes taken by the best subset selection algorithm.

```
#Finding the RMSE of the model selected using the backward selection procedure, where the
#include 2-factor interactions
pred_price = np.exp(best_bwd_reg_model.predict(test))
np.sqrt(((pred_price - test.house_price)**2).mean())
```

```
363.63365786020694
```

Note that we have not considered the null model (i.e., the model with only the intercept and no predictors) explicitly in the forward and backward stepwise algorithms. However, the null model is considered when selecting the best model. The R-squared and the adjusted R-squared for the null model is 0. So, if the adjusted R-squared of all the models with at least one predictor is negative, then the null model will be the best model.

10 Ridge regression and Lasso

Read section 6.2 of the book before using these notes.

Note that in this course, lecture notes are not sufficient, you must read the book for better understanding. Lecture notes are just implementing the concepts of the book on a dataset, but not explaining the concepts elaborately.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
```

```
trainf = pd.read_csv('./Datasets/house_feature_train.csv')
trainp = pd.read_csv('./Datasets/house_price_train.csv')
testf = pd.read_csv('./Datasets/house_feature_test.csv')
testp = pd.read_csv('./Datasets/house_price_test.csv')
train = pd.merge(trainf,trainp)
test = pd.merge(testf,testp)
train.head()
```

	house_id	house_age	distance_MRT	number_convenience_stores	latitude	longitude	house_pri
0	210	5.2	390.5684	5	24.97937	121.54245	2724.84
1	190	35.3	616.5735	8	24.97945	121.53642	1789.29
2	328	15.9	1497.7130	3	24.97003	121.51696	556.96
3	5	7.1	2175.0300	3	24.96305	121.51254	1030.41
4	412	8.1	104.8101	5	24.96674	121.54067	2756.25

Let us develop a ridge regression model to predict house price based on the five house features.

```
#Taking the log transform of house_price as house prices have a right-skewed distribution
y = np.log(train.house_price)
```

10.0.1 Standardizing the predictors

```
#Standardizing predictors so that each of them have zero mean and unit variance

#Filtering all predictors
X = train.iloc[:,1:6];

#Defining a scaler object
scaler = StandardScaler()

#The scaler object will contain the mean and variance of each column (predictor) of X.
#These values will be useful to scale test data based on the same mean and variance as obt
scaler.fit(X)

#Using the scaler object (or the values of mean and variance stored in it) to standardize
Xstd = scaler.transform(X)
```

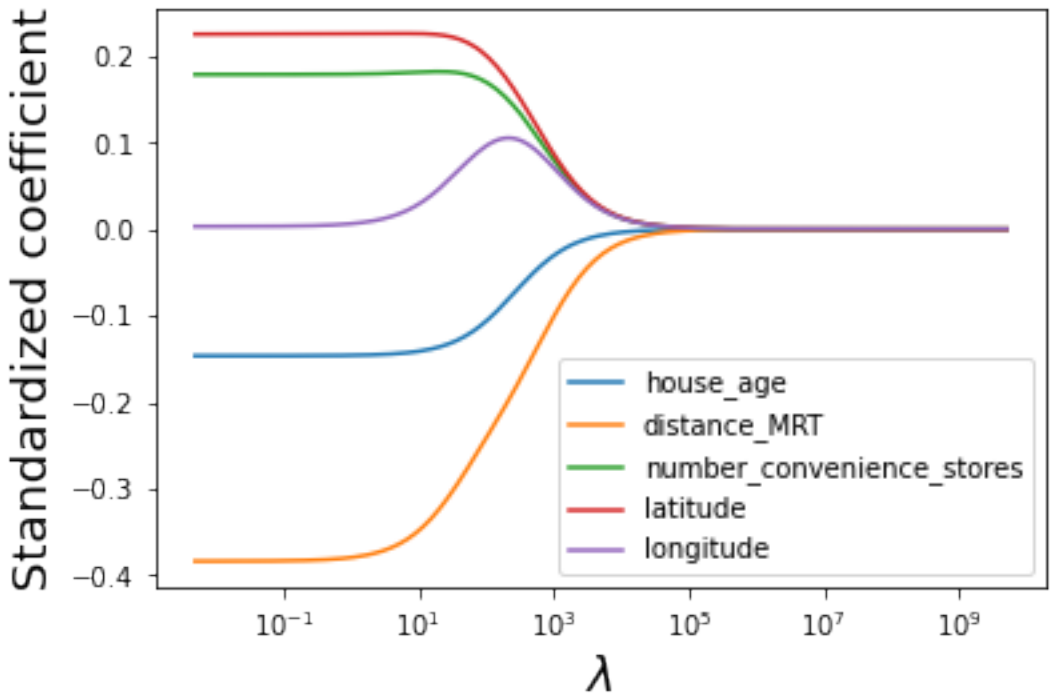
10.0.2 Optimizing the tuning parameter

```
#The tuning parameter lambda is referred as alpha in sklearn

#Creating a range of values of the tuning parameter to visualize the ridge regression coef
#for different values of the tuning parameter
alphas = 10**np.linspace(10,-2,200)*0.5
```

```
#Finding the ridge regression coefficients for increasing values of the tuning parameter
coefs = []
for a in alphas:
    ridge = Ridge(alpha = a)
    ridge.fit(Xstd, y)
    coefs.append(ridge.coef_)
```

```
#Visualizing the shrinkage in ridge regression coefficients with increasing values of the
plt.xlabel('xlabel', fontsize=18)
```

```
plt.ylabel('ylabel', fontsize=18)
plt.plot(alphas, coefs)
plt.xscale('log')
plt.xlabel('$\lambda$')
plt.ylabel('Standardized coefficient')
plt.legend(train.columns[1:6]);
```

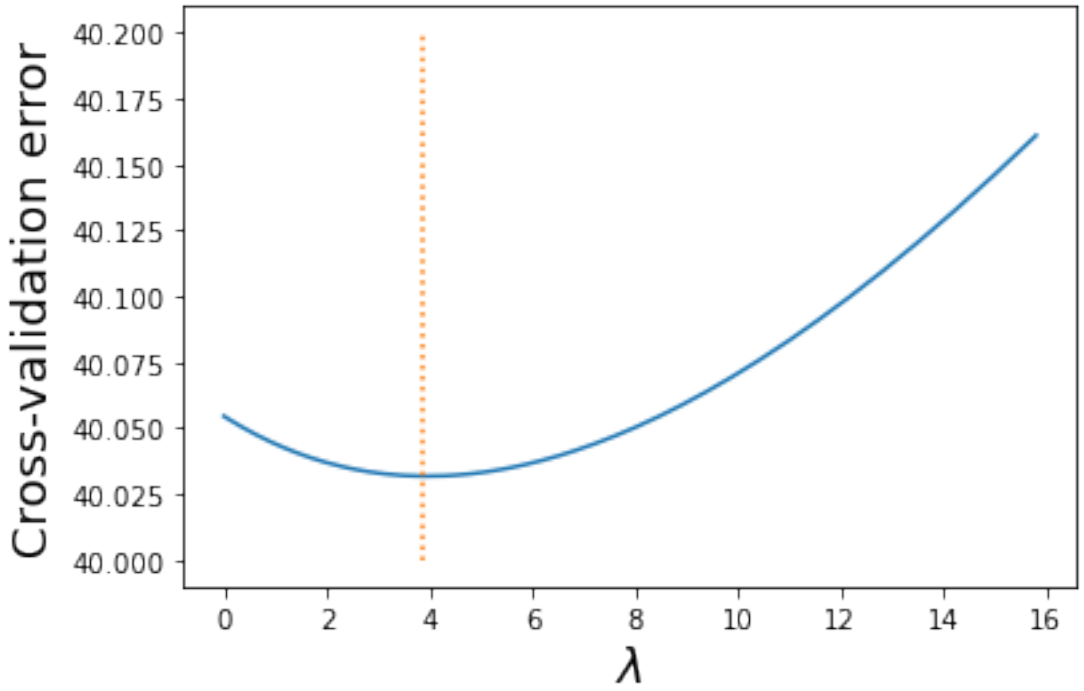



```
#Let us use cross validation to find the optimal value of the tuning parameter - lambda
#For the optimal lambda, the cross validation error will be the least

#Note that we are reducing the range of alpha so as to better visualize the minimum
alphas = 10**np.linspace(1.5,-3,200)*0.5
ridgecv = RidgeCV(alphas = alphas,store_cv_values=True)
ridgecv.fit(Xstd, y)

#Optimal value of the tuning parameter - lambda
ridgecv.alpha_
```

```
3.87629874431473
```

```
#Visualizing the LOOCV (leave one out cross validatation error vs lambda)
plt.xlabel('xlabel', fontsize=18)
plt.ylabel('ylabel', fontsize=18)
plt.plot(ridgecv.alphas,ridgecv.cv_values_.sum(axis=0))
plt.plot([ridgecv.alpha_,ridgecv.alpha_],[40,40.2],':')
plt.xlabel('$\lambda$')
plt.ylabel('Cross-validation error')
```

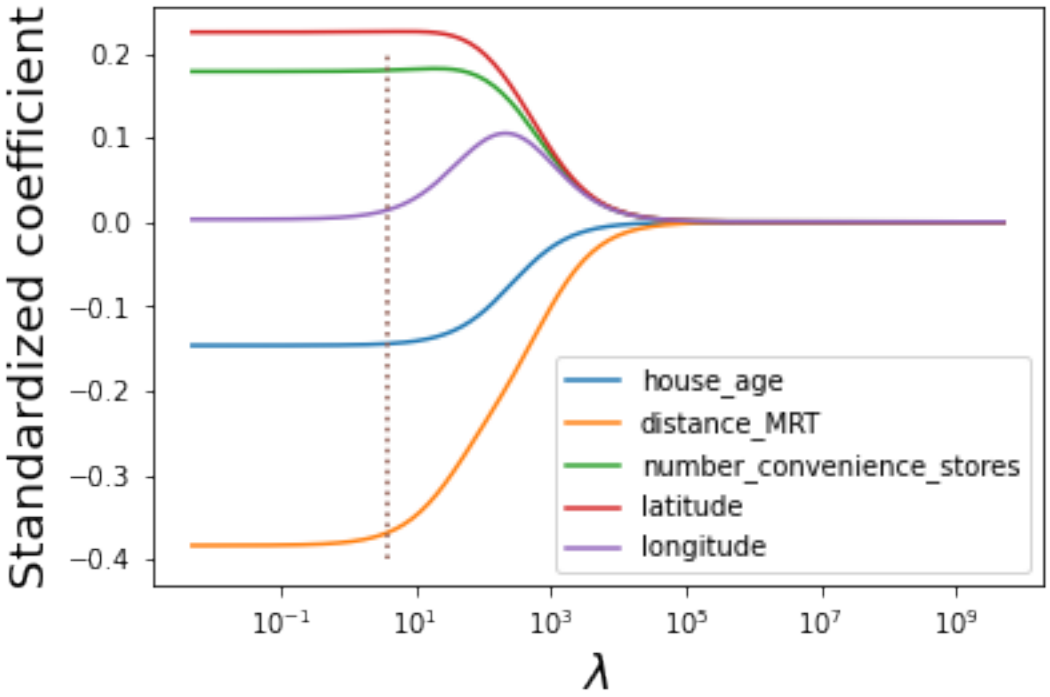
```
Text(0, 0.5, 'Cross-validation error')
```



Note that the cross validation error is minimum at the optimal value of the tuning parameter.

```
#Visualizing the shrinkage in ridge regression coefficients with increasing values of the
alphas = 10**np.linspace(10,-2,200)*0.5
plt.xlabel('xlabel', fontsize=18)
plt.ylabel('ylabel', fontsize=18)
plt.plot(alphas, coefs)
plt.plot([ridgecv.alpha_,ridgecv.alpha_],[-0.4,0.2],':')
plt.xscale('log')
```

```
plt.xlabel('$\lambda$')
plt.ylabel('Standardized coefficient')
plt.legend(train.columns[1:6]);
```



10.0.3 RMSE on test data

```
#Test dataset
Xtest = test.iloc[:,1:6]

#Standardizing test data
Xtest_std = scaler.transform(Xtest)
```

```
#Using the developed ridge regression model to predict on test data
ridge = Ridge(alpha = ridgecv.alpha_)
ridge.fit(Xstd, y)
pred=ridge.predict(Xtest_std)
```

```
#RMSE on test data
np.sqrt(((np.exp(pred)-test.house_price)**2).mean())
```

```
405.6227485138042
```

Note that the RMSE is similar to the one obtained using least squares regression on all the five predictors. This is because the coefficients were required to shrink very slightly for the best ridge regression fit. This may happen when we have a low number of predictors, where most of them are significant. Ridge regression is likely to perform better than least squares in case of a large number of predictors, where an OLS model will be prone to overfitting.

10.0.4 Model coefficients & R-squared

```
#Checking the coefficients of the ridge regression model
ridge.coef_
```

```
array([-0.1444778 , -0.36856553,  0.17986479,  0.22566444,  0.01413125])
```

Note that none of the coefficients are shrunk to zero. The coefficient of `longitude` is smaller than the rest, but not zero.

```
#R-squared on train data for the ridge regression model
r2_score(ridge.predict(Xstd),y)
```

```
0.6994484432136066
```

```
#R-squared on test data for the ridge regression model
r2_score(pred,np.log(test.house_price))
```

```
0.7573027646359806
```

10.1 Lasso

Let us develop a lasso model to predict house price based on the five house features.

10.1.1 Standardizing the predictors

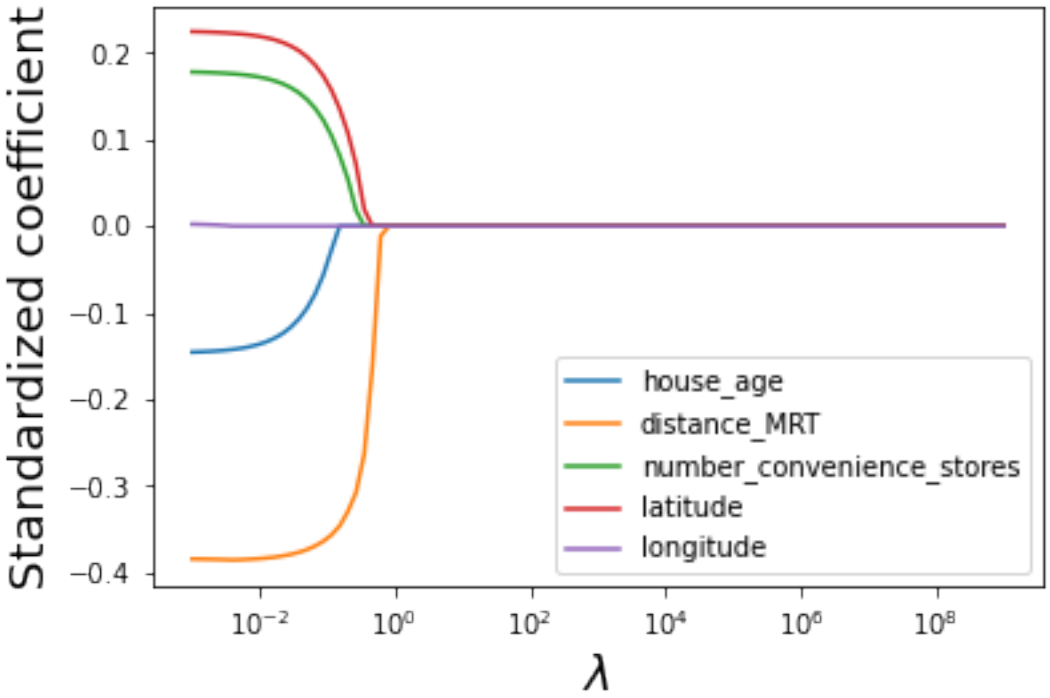
We have already standardized the predictors in the previous section. The standardized predictors are the NumPy array object `Xstd`.

10.1.2 Optimizing the tuning parameter

```
#Creating a range of values of the tuning parameter to visualize the lasso coefficients
#for different values of the tuning parameter
alphas = 10**np.linspace(10,-2,100)*0.1
```

```
#Finding the lasso coefficients for increasing values of the tuning parameter
lasso = Lasso(max_iter = 10000)
coefs = []

for a in alphas:
    lasso.set_params(alpha=a)
    lasso.fit(Xstd, y)
    coefs.append(lasso.coef_)
```

```
#Visualizing the shrinkage in lasso coefficients with increasing values of the tuning para
plt.xlabel('xlabel', fontsize=18)
plt.ylabel('ylabel', fontsize=18)
plt.plot(alphas, coefs)
plt.xscale('log')
plt.xlabel('$\lambda$')
plt.ylabel('Standardized coefficient')
plt.legend(train.columns[1:6]);
```



Note that lasso performs variable selection. For certain values of lambda, some of the predictor coefficients are zero, while others are non-zero. This is different than ridge regression, which only shrinks the coefficients, but doesn't do variable selection.

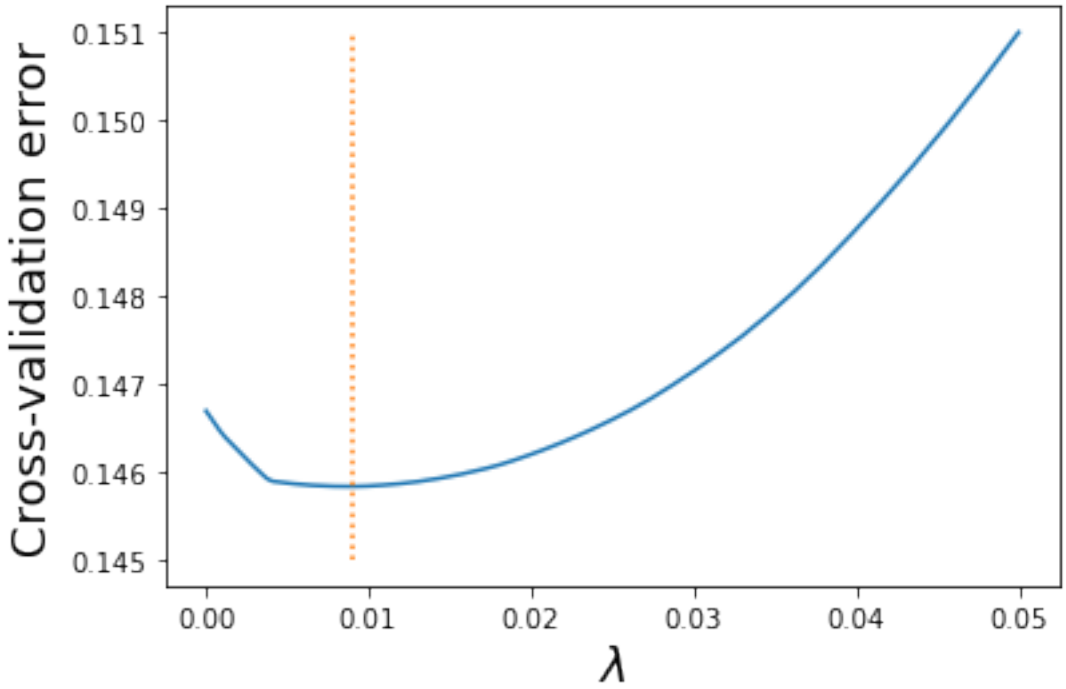
```
#Let us use cross validation to find the optimal value of the tuning parameter - lambda
#For the optimal lambda, the cross validation error will be the least

#Note that we are reducing the range of alpha so as to better visualize the minimum
alphas = 10**np.linspace(-1,-5,200)*0.5
lassocv = LassoCV(alphas = alphas, cv = 10, max_iter = 100000)
lassocv.fit(Xstd, y)

#Optimal value of the tuning parameter - lamda
lassocv.alpha_
```

```
0.009020932046960358
```

```
#Visualizing the LOOCV (leave one out cross validatation error vs lambda)
plt.xlabel('xlabel', fontsize=18)
```

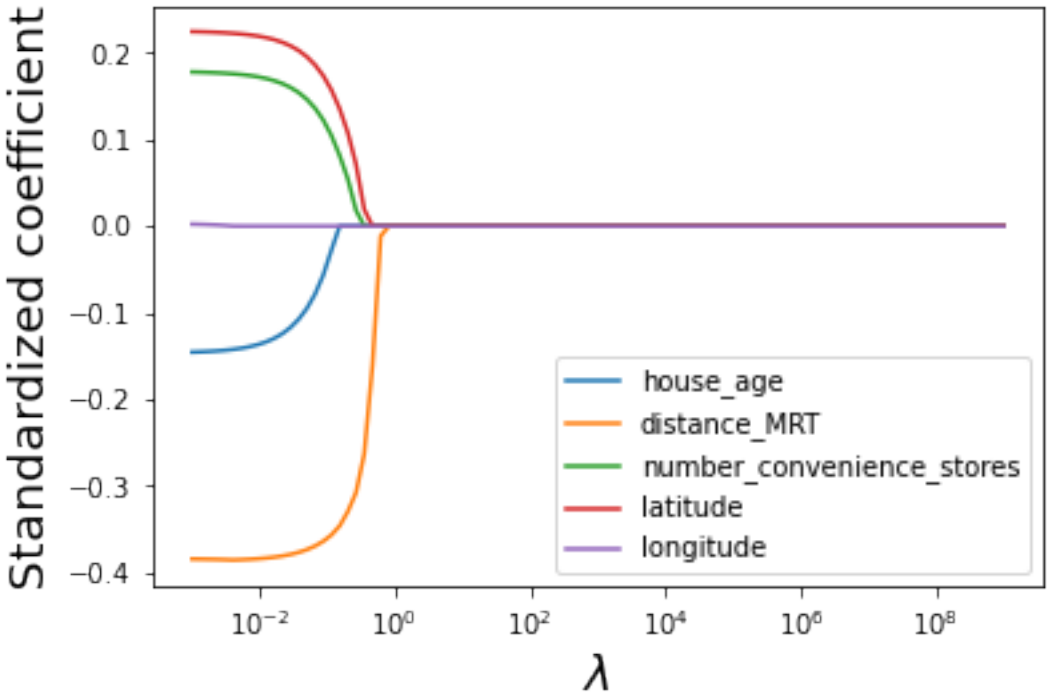
```
plt.ylabel('ylabel', fontsize=18)
plt.plot(lassocv.alphas_,lassocv.mse_path_.mean(axis=1))
plt.plot([lassocv.alpha_,lassocv.alpha_],[0.145,0.151],':')
plt.xlabel('$\lambda$')
plt.ylabel('Cross-validation error')
```

```
Text(0, 0.5, 'Cross-validation error')
```



The 10-fold cross validation error minimizes at lambda = 0.009.

```
#Visualizing the shrinkage in lasso coefficients with increasing values of the tuning para
alphas = 10**np.linspace(10,-2,100)*0.1
plt.xlabel('xlabel', fontsize=18)
plt.ylabel('ylabel', fontsize=18)
plt.plot(alphas, coefs)
plt.xscale('log')
plt.xlabel('$\lambda$')
plt.ylabel('Standardized coefficient')
plt.legend(train.columns[1:6]);
```



10.1.3 RMSE on test data

```
#Using the developed lasso model to predict on test data
lasso = Lasso(alpha = lassocv.alpha_)
lasso.fit(Xstd, y)
pred=lasso.predict(Xtest_std)
```

```
#RMSE on test data
np.sqrt(((np.exp(pred)-test.house_price)**2).mean())
```

```
400.77289943396534
```

10.1.4 Model coefficients & R-squared

```
#Checking the coefficients of the lasso model
lasso.coef_
```

```
array([-0.13720237, -0.38405197,  0.17252859,  0.21949239,  0.        ])
```

Note that the coefficient of `longitude` is shrunk to zero. Lasso performs variable selection.

```
#R-squared on train data for the lasso model
r2_score(lasso.predict(Xstd),y)
```

```
0.692606850601813
```

```
#R-squared on test data for the lasso model
r2_score(pred,np.log(test.house_price))
```

```
0.7524177148260849
```

A Assignment A

1. You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and not as a group activity.
2. Do not write your name on the assignment.
3. Write your code in the *Code* cells and your answer in the *Markdown* cells of the Jupyter notebook. Ensure that the solution is written neatly enough to understand and grade.
4. Use Quarto to print the *.ipynb* file as HTML. You will need to open the command prompt, navigate to the directory containing the file, and use the command: `quarto render filename.ipynb --to html`. Submit the HTML file.
5. The assignment is worth 100 points, and is due on **Tuesday, 17th January 2023 at 11:59 pm**.
6. There is a **bonus** question worth 5 points.
7. **Five points are for properly formatting the assignment**. The breakdown is as follows:

- Must be an HTML file rendered using Quarto (1 pt); *If you have a Quarto issue, you must mention the issue & quote the error you get when rendering using Quarto in the comments section of Canvas, and submit the ipynb file.*
- No name can be written on the assignment, nor can there be any indicator of the student's identity—e.g., printouts of the working directory should not be included in the final submission (1 pt).
- There aren't excessively long outputs of extraneous information (e.g. no printouts of entire data frames without good reason, there aren't long printouts of which iteration a loop is on, there aren't long sections of commented-out code, etc.) (1 pt).
- Final answers of each question are written in Markdown cells (1 pt).
- There is no piece of unnecessary / redundant code, and no unnecessary / redundant text (1 pt).

8. The maximum possible score in the assigment is 95 + 5 (formatting) + 5 (bonus question) = 105 out of 100. There is no partial credit for the bonus question.

A.1 Regression vs Classification; Prediction vs Inference

Explain (1) whether each scenario is a classification or regression problem, and (2) whether we are most interested in inference or prediction. Answers to both parts must be supported by a justification.

A.1.1

Consider a company that is interested in conducting a marketing campaign. The goal is to identify individuals who are likely to respond positively to a marketing campaign, based on observations of demographic variables *(such as age, gender, income, etc.)* measured on each individual.

(2+2 points)

A.1.2

Consider that the company mentioned in the previous question is interested in understanding the impact of advertising promotions in different media types on the company sales. For example, the company is interested in the question, *'how large of an increase in sales is associated with a given increase in radio vis-a-vis TV advertising?'*

(2+2 points)

A.1.3

Consider a company selling furniture is interested in the finding the association between demographic characterisitcs of customers (such as age, gender, income, etc.) and their probability of purchase of a particular company product.

(2+2 points)

A.1.4

We are interested in predicting the % change in the USD/Euro exchange rate in relation to the weekly changes in the world stock markets. Hence we collect weekly data for all of 2022. For each week we record the % change in the USD/Euro, the % change in the US market, the % change in the British market, and the % change in the German market.

(2+2 points)

A.2 RMSE vs MAE

A.2.1

Describe a regression problem, where it will be more appropriate to assess the model accuracy using the root mean squared error (RMSE) metric as compared to the mean absolute error (MAE) metric.

Note: Don't use the examples presented in class

(4 points)

A.2.2

Describe a regression problem, where it will be more appropriate to assess the model accuracy using the mean absolute error (MAE) metric as compared to the root mean squared error (RMSE) metric.

Note: Don't use the examples presented in class

(4 points)

A.3 FNR vs FPR

A.3.1

A classification model is developed to predict those customers who will respond positively to a company's tele-marketing campaign. All those customers that are predicted to respond positively to the campaign will be called by phone to buy the product being marketed. If the customer being called purchases the product ($y = 1$), the company will get a profit of \$100. On the other hand, if they are called and they don't purchase ($y = 0$), the company will have a loss of \$1. Among FPR (False positive rate) and FNR (False negative rate), which metric is more important to be minimized to reduce the loss associated with misclassification? Justify your answer.

In your justification, you must clearly interpret False Negatives (FN) and False Postives (FP) first.

Assumption: Assume that based on the past marketing campaigns, around 50% of the customers will actually respond positively to the campaign.

(4 points)

A.3.2

Can the answer to the previous question change if the assumption stated in the question is false? Justify your answer.

(6 points)

A.4 Petrol consumption

Read the dataset *petrol_consumption_train.csv*. It contains the following five columns:

`Petrol_tax`: Petrol tax (cents per gallon)

`Per_capita_income`: Average income (dollars)

`Paved_highways`: Paved Highways (miles)

`Prop_license`: Proportion of population with driver's licenses

`Petrol_consumption`: Consumption of petrol (millions of gallons)

A.4.1

Make a pairwise plot of all the variables in the dataset. Which variable seems to have the highest linear correlation with `Petrol_consumption`? Let this variable be predictor P. *Note: If you cannot figure out P by looking at the visualization, you may find the pairwise linear correlation coefficient to identify P.*

(4 points)

A.4.2

Fit a simple linear regression model to predict `Petrol_consumption` based on predictor P (identified in the previous part). Print the model summary.

(4 points)

A.4.3

Interpret the coefficient of P. What is the increase in petrol consumption for an increase of 0.05 in P?

(2+2 points)

A.4.4

Does petrol consumption have a statistically significant relationship with the predictor P? Justify your answer.

(4 points)

A.4.5

What is the R-squared? Interpret its value.

(4 points)

A.4.6

Use the model developed above to estimate the petrol consumption for a state in which 50% of the population has a driver's license. What are the confidence and prediction intervals for your estimate? Which interval includes the irreducible error?

(4+3+3+2 = 12 points)

A.4.7

Use the model developed above to estimate the petrol consumption for a state in which 10% of the population has a driver's license. Are you getting a reasonable estimate? Why or why not?

(5 points)

A.4.8

What is the residual standard error of the model?

(4 points)

A.4.9

Using the model developed above, predict the petrol consumption for the observations in *petrol_consumption_test.csv*. Find the RMSE (Root mean squared error). Include the units of RMSE in your answer.

(5 points)

A.4.10

Based on the answers to the previous two questions, do you think the model is overfitting? Justify your answer.

(4 points)

Make a scatterplot of `Petrol_consumption` vs `Prop_license` using *petrol_consumption_test.csv*. Over the scatterplot, plot the regression line, the prediction interval, and the confidence interval. Distinguish the regression line, prediction interval lines, and confidence interval lines with the following colors. Include the legend as well.

- Regression line: red
- Confidence interval lines: blue
- Prediction interval lines: green

(4 points)

Among the confidence and prediction intervals, which interval is wider, and why?

(1+2 points)

A.4.11

Find the correlation between `Petrol_consumption` and the rest of the variables in *petrol_consumption_train.csv*. Based on the correlations, a simple linear regression model with which predictor will have the least R-squared value for predicting `Petrol_consumption`. Don't develop any linear regression models.

(4 points)

Bonus point question

(5 points - no partial credit)

A.4.12

Fit a simple linear regression model to predict `Petrol_consumption` based on predictor P, but without an intercept term.

(you must answer this correctly to qualify for earning bonus points)

A.4.13

Estimate the petrol consumption for the observations in *petrol_consumption_test.csv* using the model in developed in the previous question. Find the RMSE.

(you must answer this correctly to qualify for earning bonus points)

A.4.14

The RMSE for the models with and without the intercept are similar, which indicates that both models are almost equally good. However, the R-squared for the model without intercept is much higher than the R-squared for the model with the intercept. Why? Justify your answer.

(5 points)

B Assignment B

1. You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and not as a group activity.
2. Do not write your name on the assignment.
3. Write your code in the *Code* cells and your answer in the *Markdown* cells of the Jupyter notebook. Ensure that the solution is written neatly enough to understand and grade.
4. Use Quarto to print the *.ipynb* file as HTML. You will need to open the command prompt, navigate to the directory containing the file, and use the command: `quarto render filename.ipynb --to html`. Submit the HTML file.
5. The assignment is worth 100 points, and is due on **Thursday, 26th January 2023 at 11:59 pm**.
6. **Five points are properly formatting the assignment**. The breakdown is as follows:

- Must be an HTML file rendered using Quarto (1 pt). *If you have a Quarto issue, you must mention the issue & quote the error you get when rendering using Quarto in the comments section of Canvas, and submit the ipynb file.*
- No name can be written on the assignment, nor can there be any indicator of the student's identity—e.g. printouts of the working directory should not be included in the final submission (1 pt)
- There aren't excessively long outputs of extraneous information (e.g. no printouts of entire data frames without good reason, there aren't long printouts of which iteration a loop is on, there aren't long sections of commented-out code, etc.) (1 pt)
- Final answers of each question are written in Markdown cells (1 pt).
- There is no piece of unnecessary / redundant code, and no unnecessary / redundant text (1 pt)

B.1 Multiple linear regression

A study was conducted on 97 men with prostate cancer who were due to receive a radical prostatectomy. The dataset *prostate.csv* contains data on 9 measurements made on these 97 men. The description of variables can be found here:

B.1.1 Training MLR

Fit a linear regression model with `lpsa` as the response and all the other variables as predictors. Write down the equation to predict `lpsa` based on the other eight variables.

(2+2 points)

B.1.2 Model significance

Is the overall regression significant at 5% level? Justify your answer.

(2 points)

B.1.3 Coefficient interpretation

Interpret the coefficient of `svi`.

(2 points)

B.1.4 Variable significance

Report the p-values for `gleason` and `age`. What do you conclude about the significance of these variables?

(2+2 points)

B.1.5 Variable significance from confidence interval

What is the 95% confidence interval for the coefficient of `age`? Can you conclude anything about its significance based on the confidence interval?

(2+2 points)

B.1.6 p-value

Fit a simple linear regression on `lpsa` against `gleason`. What is the p-value for `gleason`?

(1+1 points)

B.1.7 Predictor significance in presence / absence of other predictors

Is the predictor `gleason` statistically significant in the model developed in the previous question *(B.1.6)*?

Was `gleason` statistically significant in the model developed in the first question *(B.1.1)* with multiple predictors?

Did the statistical significance of `gleason` change in the absence of other predictors? Why or why not?

(1+1+4 points)

B.1.8 Prediction

Predict `lpsa` of a 65-year old man with `lcavol` = 1.35, `lweight` = 3.65, `lbph` = 0.1, `svi` = 0.22, `lcp` = -0.18, `gleason` = 6.75, and `pgg45` = 25 and find 95% prediction intervals.

(2 points)

B.1.9 Variable selection

Find the largest subset of predictors in the model developed in the first question *(B.1.1)*, such that their coefficients are zero, i.e., none of the predictors in the subset are statistically significant.

Does the model R-squared change a lot if you remove the set of predictors identifed above from the model in the first question *(B.1.1)*?

Hint: You may use the `f_test()` method to test hypotheses.

(4+1 points)

B.2 Using MLR coefficients and variable transformation

The dataset *infmort.csv* gives the infant mortality of different countries in the world. The column `mortality` contains the infant mortality in deaths per 1000 births.

B.2.1 Data visualisation

Make the following plots:

1. a boxplot of log(`mortality`) against `region` *(note that a plot of log(`mortality`) against `region` better distinguishes the mortality among regions as compared to a plot of `mortality` against `region`*,
2. a boxplot of `income` against `region`, and
3. a scatter plot of `mortality` against `income`.

What trends do you see in these plots? *Mention the trend separately for each plot.*

(3+2 points)

B.2.2 Removing effect of predictor from response

Europe seems to have the lowest infant mortality, but it also has the highest per capita annual income. We want to see if Europe still has the lowest mortality if we remove the effect of income from the mortality. We will answer this question with the following steps.

B.2.2.1 Variable transformation

Plot:

1. `mortality` against `income`,
2. log(`mortality`) against `income`,
3. `mortality` against log(`income`), and
4. log(`mortality`) against log(`income`).

Based on the plots, postulate an appropriate model to predict mortality as a function of income. *Print the model summary.*

(2+4 points)

B.2.2.2 Model update

Update the model developed in the previous question by adding `region` as a predictor. Print the model summary.

(2 points)

Use the model developed in the previous question to compute `adjusted_mortality` for each observation in the data, where adjusted mortality is the mortality after removing the estimated effect of income. Make a boxplot of log(`adjusted_mortality`) against `region`.

(4+2 points)

B.2.3 Data visualisation after removing effect of predictor from response

From the plot in the previous question:

1. Does Europe still seem to have the lowest mortality as compared to other regions after removing the effect of income from mortality?
2. After adjusting for income, is there any change in the mortality comparison among different regions. Compare the plot developed in the previous question to the plot of `log(mortality)` against `region` developed earlier *(B.2.1)* to answer this question.

Hint: Do any African / Asian / American countries seem to do better than all the European countries with regard to mortality after adjusting for income?

(1+3 points)

B.3 Variable transformations and interactions

The dataset *soc_ind.csv* contains the GDP per capita of some countries along with several social indicators.

B.3.1 Training SLR

For a simple linear regression model predicting `gdpPerCapita`. Which predictor will provide the best model fit *(ignore categorical predictors)*? Let that predictor be P.

(2 points)

B.3.2 Linearity in relationship

Make a scatterplot of `gdpPerCapita` vs P. Does the relationship between `gdpPerCapita` and P seem linear or non-linear?

(1 + 2 points)

B.3.3 Variable transformation

If the relationship identified in the previous question is non-linear, identify and include transformation(s) of the predictor P in the model to improve the model fit.

Mention the predictors of the transformed model, and report the change in the R-squared value of the transformed model as compared to the simple linear regression model with only P.

(4+4 points)

B.3.4 Model visualisation with transformed predictor

Plot the regression curve of the transformed model *(developed in the previous question)* over the scatterplot in (b) to visualize model fit. Also make the regression line of the simple linear regression model with only P on the same plot.

(3 + 1 points)

B.3.5 Training MLR with qualitative predictor

Develop a model to predict `gdpPerCapita` with P and `continent` as predictors.

1. Interpert the intercept term.
2. For a given value of P, are there any continents that **do not** have a signficant difference between their mean `gdpPerCapita` and that of Africa? If yes, then which ones, and why? If no, then why not? Consider a significance level of 5%.

(4 + 4 points)

B.3.6 Variable interaction

The model developed in the previous question has a limitation. It assumes that the increase in mean `gdpPerCapita` with a unit increase in P does not depend on the `continent`.

1. Eliminate this limitation by including interaction of `continent` with P in the model developed in the previous question. Print the model summary of the model with interactions.
2. Interpret the coefficient of any one of the interaction terms.

(4 + 4 points)

B.3.7 Model visualisation with qualitative predictor

Use the model developed in the previous question to plot the regression lines for Africa, Asia, and Europe. Put `gdpPerCapita` on the vertical axis and P on the horizontal axis. Use a legend to distinguish among the regression lines of the three continents.

(4 points)

B.3.8 Model interpretation

Based on the plot develop in the previous question, which continent has the highest increase in mean `gdpPerCapita` for a unit increase in P, and which one has the least? Justify your answer.

(2+2 points)

C Assignment C

1. You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and not as a group activity.
2. Do not write your name on the assignment.
3. Write your code in the *Code* cells and your answer in the *Markdown* cells of the Jupyter notebook. Ensure that the solution is written neatly enough to understand and grade.
4. Use Quarto to print the *.ipynb* file as HTML. You will need to open the command prompt, navigate to the directory containing the file, and use the command: `quarto render filename.ipynb --to html`. Submit the HTML file.
5. The assignment is worth 100 points, and is due on **Thursday, 9th February 2023 at 11:59 pm**.
6. **Five points are properly formatting the assignment**. The breakdown is as follows:

- Must be an HTML file rendered using Quarto (1 pt). *If you have a Quarto issue, you must mention the issue & quote the error you get when rendering using Quarto in the comments section of Canvas, and submit the ipynb file.*
- No name can be written on the assignment, nor can there be any indicator of the student's identity—e.g. printouts of the working directory should not be included in the final submission (1 pt)
- There aren't excessively long outputs of extraneous information (e.g. no printouts of entire data frames without good reason, there aren't long printouts of which iteration a loop is on, there aren't long sections of commented-out code, etc.) (1 pt)
- Final answers of each question are written in Markdown cells (1 pt).
- There is no piece of unnecessary / redundant code, and no unnecessary / redundant text (1 pt)

C.1 Model assumptions

C.1.1

Using *house_feature_train.csv* and *house_price_train.csv*, fit a multiple linear regression model without transformation to predict `house_price` based on `distance_MRT`, `latitude`, and `longitude`, `house_age`, and `number_convenience_stores`. Print the model summary. What is the model R^2?

(1 + 1 points)

C.1.2

Obtain the residuals and plot them separately against fitted values and each of the five feature variables. Make one plot including the 6 subplots.

(4 points)

C.1.3

Comment on the plot of residuals against fitted values. Does the model violate the assumption of linearity? Does the model violate the constant variance assumption?

(2 + 2 points)

C.1.4

Comment on the plot of residuals against the predictor variables. On the basis of these plots, should any further modifications of the regression model be attempted?

(5 points)

C.1.5

Calculate the RMSE using the test datasets for the model constructed in the first question. The test datasets are *house_feature_test.csv* and *house_price_test.csv*.

(2 points)

C.1.6

Using appropriate transformation(s) and/or variable interaction(s), update the model to obtain a model that has a R^2 of at least 80%, and a RMSE (Root mean squared error) of at max \$350k on test data.

Print the model summary and report the R^2, and RMSE on test data. Note:

1. House prices are provided in thousands of dollars. A value of 556 in the `house_price` column indicates a house price of \$556k.
2. The test datasets are *house_feature_test.csv* and *house_price_test.csv*.
3. R^2 is computed on training data, and RMSE is computed on test data.
4. You must proceed logically, i.e., **justify every transformation** that you introduce into the model to improve it. If you are introducing **interactions**, there should be some rationale behind including only certain interactions in the model, unless you are including all possible interactions.

(12 points for achieving the objectives + 8 points for justifications)

C.1.7

Are the assumptions of linearity and constant variance of errors satisfied in the model developed in the previous question? Make a scatterplot between the residuals and fitted values and use it to answer the question.

(4 points)

C.2 Multicollinearity and Outliers

The datasets *Austin_Affordable_Housing_Train.csv* and *Austin_Affordable_Housing_Test.csv* provide data on housing development projects that have received funding from the Affordable Housing Development Fund in Austin, Texas. The city provides property developers with tax credits and other forms of funding in exchange for agreements to set housing prices (e.g. rent) below market rate.

Each row represents a housing development in Austin. Variables include the amount (USD) provided by the city, the status of the housing project, the number of housing units, the period of affordability, and more.

Let's say that you're hired by the city as a consultant to work with subject matter experts in their Housing and Planning Department.

General Hint: For written sections, writing "it depends" (along with an explanation) often characterizes a good answer.

Note for Grading Team: Written answers should be given full credit as long as they're thoughtful answers that address the question fully, base findings on relevant data/results, and align with the relevant regression theory/thinking. Many questions don't have a single right answer and/or depend on context that isn't provided here.

C.2.1

Suppose you run the code `status_vars = pd.get_dummies(housing_dataframe["Status"])`, append the columns of `status_vars` to your original data frame, and use the columns as predictors in a linear regression model. What potential problem would you likely be introducing into the model? How could it affect your results?

(4 points)

C.2.2

Suppose that a subject matter expert recommends using the variables `Total_Units`, `Total_Affordable_Units`, `Total_Accessible_Units`, and `Market_Rate_Units` as predictors in your model. From a regression modeling standpoint, does this sound advisable? Produce metrics to quantify the potential impact of including the four predictors in a model. Interpret at least one of the metrics you provide, both statistically and in the context of the problem.

(4 points)

C.2.3

Say that the subject matter expert agrees to use `Total_Affordable_Units`, `Affordability_Expiration_Year`, and `Units_Under_50_Percent_MFI` as predictors for `City_Amount`. Fit the appropriate model (without transformations). Then interpret the results associated with `Total_Affordable_Units`, and comment on the overall model fit.

(4 points)

C.2.4

Using visualizations, investigate whether the model you fit in in the previous question yields outlying observations. What count and proportion of observations would you classify as outliers?

Note: Show separate plots for both - residuals and studentized residuals. However, consider studentized residuals when identifying outliers.

(4 points)

C.2.5

Based on your results in the previous question, would you choose to remove outlying observations? Why or why not?

(4 points)

C.2.6

Consider a scenario in which the model will be used by property owners seeking to predict the amount of money they may receive from the city of Austin. How would this change, support, or complicate your answer in the previous question, if at all?

(3 points)

C.2.7

Say that the model will be used by a team of sociologists seeking statistical evidence at the $\alpha = 0.01$ significance level that a property's affordability expiration year has an effect on the amount of money issued by the city of Austin? How would this change, support, or complicate your answer in C.2.5, if at all?

(3 points)

C.2.8

Determine whether the model you fit in C.2.3 contains any high-leverage points. Produce a visualization, then report the count and proportions of observations that are high-leverage *(define an observation as "high-leverage" if its leverage is greater than four times the average leverage of all observations).*

(4 points)

C.2.9

Based on your results in the previous question, would you choose to remove high-leverage observations? Why or why not?

(3 points)

C.2.10

Identify and remove any influential points from the training data and refit the model. How does removing the influential points affect the model, if at all?

Think about using the model summary, and use the test data provided.

(6 points)

C.3 Autocorrelation

Refer to the autocorrelation example in the class notes. Predict the power consumption for each hour of each day of the year 2020. For predicting a power consumption on a particular hour of a day, use all the data you have until the previous day. However, don't use any data of the day on which you are making the predictions. For example, for making 24 predictions for each hour of 4th April, 2020, use all the data upto 3rd April 2020. Make the predictions using four different models:

1. Model with only `temp_hot` and `temp_cold` as the predictors
2. Model including one day lag of power as a predictor in addition to the predictors in model (1)
3. Model including one week lag of power as a predictor in addition to the predictors in model (2)
4. Model including two weeks lag of power as a predictor in addition to the predictors in model (3)

For each model:

1. Report the RMSE for the predicted power in 2020. You should have 366 x 24 = 8784 predicted values of power for each model.
2. Make a scatterplot of predicted power vs actual power *(use color = 'orange')*. Plot the line x = y over the scatterplot *(use color = 'blue')*.

Which model makes the most accurate predictions?

(4 points for developing the models + 4 points for computing the predictions + 4 points for computing the RMSEs + 2 points for the visualizations + 1 point for identifying the most accurate model)

D Assignment D

1. You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and not as a group activity.
2. Do not write your name on the assignment.
3. Write your code in the *Code* cells and your answer in the *Markdown* cells of the Jupyter notebook. Ensure that the solution is written neatly enough to understand and grade.
4. Use Quarto to print the *.ipynb* file as HTML. You will need to open the command prompt, navigate to the directory containing the file, and use the command: `quarto render filename.ipynb --to html`. Submit the HTML file.
5. The assignment is worth 100 points, and is due on **Tuesday, 21st February 2023 at 11:59 pm**.
6. **Five points are properly formatting the assignment**. The breakdown is as follows:

- Must be an HTML file rendered using Quarto (1 pt). *If you have a Quarto issue, you must mention the issue & quote the error you get when rendering using Quarto in the comments section of Canvas, and submit the ipynb file.*
- No name can be written on the assignment, nor can there be any indicator of the student's identity—e.g. printouts of the working directory should not be included in the final submission (1 pt)
- There aren't excessively long outputs of extraneous information (e.g. no printouts of entire data frames without good reason, there aren't long printouts of which iteration a loop is on, there aren't long sections of commented-out code, etc.) (1 pt)
- Final answers of each question are written in Markdown cells (1 pt).
- There is no piece of unnecessary / redundant code, and no unnecessary / redundant text (1 pt)

Data description

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls, where bank clients were called to subscribe for a term deposit.

There is one train data - *train.csv*, which you will use to develop a model. There are two test datasets - *test1.csv* and *test2.csv*, which you will use to test your model. Each dataset has the following attributes about the clients called in the marketing campaign:

1. `age`: Age of the client
2. `education`: Education level of the client
3. `day`: Day of the month the call is made
4. `month`: Month of the call
5. `y`: did the client subscribe to a term deposit?
6. `duration`: Call duration, in seconds. This attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the `duration` is not known before a call is performed. Also, after the end of the call `y` is obviously known. Thus, this input should only be included for inference purposes and should be discarded if the intention is to have a realistic predictive model.

(Raw data source: Source. Do not use the raw data source for this assingment. It is just for reference.)

Instructions / suggestions for answering questions

(1) **Instruction:** Use *train.csv* for all questions, unless otherwise stated.

(2) **Suggestion 1:** You may use the functions in the class notes for printing the confusion matrix and the overall classification accuracy based on test / train data.

(3) **Suggestion 2:**: If you make variable transformations, you will need to do it for all the three datasets. Your code will be a bit concise if you make a function containing all the transformations, and then call it for the training and the two test datasets. You can put this function in the beginning of the code and keep adding transformations to it as you proceed with the assignment. You may need transformations in questions (1) and (13).

D.1 Probability of response vs call duration

Read the datasets. Make an appropriate visualization to visualize how the proportion of clients subscribing to a term deposit change with increasing call duration.

(4 points)

Hints:

1. Bin `duration` to create `duration_binned`. Group the data to find the fraction of clients responding positively to the marketing campaign for each bin in `duration_binned`. Make a lineplot of percentage of clients subscribing to a term deposit vs `duration_binned`, where the bins in `duration_binned` are arranged in increasing order of duration.
2. You may choose an appropriate number of bins & type of binning that helps you visualize well.
3. You may also think of other ways of visualization. You don't need to stick with this one.

D.2 Predictor `duration`

Based on the plot in D.1, comment whether `duration` seems to be a useful variable to predict if the client will subscribe to a term deposit.

(1 point)

D.3 Model based on `duration`

Develop a logistic regression model to predict if the client subscribed to a term deposit based on call `duration`. Use the model to make a lineplot showing the probability of the client subscribing to a term deposit based on call `duration`.

(3 points)

Note

Answer questions D.4 to D.11 based on the regression model developed in D.3.

D.4 Model significance

Is the regression model in statistically significant? Justify your answer.

(1 point for code, 1 point for answer)

D.5 Subscription probability in 5 minutes

What is the probability that the client subscribes to a term deposit with a 5-minute marketing call? Note that the call `duration` in data is given in *seconds*.

(2 points)

D.6 Call duration for subscription

What is the minimum call duration (in minutes) for which a client has a 95% or higher chance of subscribing to a term deposit?

(4 points)

D.7 Maximum call duration

What is the maximum call duration (in minutes) in which a client refused to subscribe to a term deposit? What was the probability of the client subscribing to the term deposit in that call?

(4 points)

D.8 Percent increase in odds

What is the percentage increase in the odds of a client subscribing to a term deposit when the call `duration` increases by a minute?

(4 points)

D.9 Doubling the subscription odds

How much must the call `duration` increase (in minutes) so that it doubles the odds of the client subscribing to a term deposit.

(3 points)

D.10 Classification accuracy

What is minimum overall classification accuracy of the model among the classification accuracies on *train.csv*, *test1.csv* and *test2.csv*? Consider a threshold of 30% when classifying observations.

(2 + 1 + 1 points)

D.11 Recall

What is the minimum *Recall* of the model among the *Recall* performance on *train.csv*, *test1.csv* and *test2.csv*? Consider a decision threshold probability of 30% when classifying observations.

Here, *Recall* is the proportion of clients predicted to subscribe to a term deposit among those who actually subscribed.

(3 points)

D.12 Subscription probability based on `age` and `education`

Develop a logistic regression model to predict the probability of a client subscribing to a term deposit based on `age`, `education` and the two-factor interaction between `age` and `education`. Based on the model, answer:

a. People with which type of `education` *(primary / secondary / tertiary / unknown)* have the highest percentage increase in odds of subscribing to a term deposit with a unit increase in `age`? Justify your answer.

b. What is the percentage increase in odds of a person subscribing to a term deposit for a unit increase in `age`, if the person has *tertiary* `education`.

c. What is the percentage increase in odds of a person subscribing to a term deposit for a unit increase in `age`, if the person has *primary* `education`.

(1 point for developing the model, 3 points for (a), 3 points for (b), 3 points for (c))

D.13 Model development

Develop a logistic regression model *(using train.csv)* to predict the probability of a client subscribing to a term deposit based on `age`, `education`, `day` and `month`. The model must have:

a. Minimum overall classification accuracy of 75% among the classification accuracies on *train.csv*, *test1.csv* and *test2.csv*.

b. Minimum recall of 50% among the recall performance on *train.csv*, *test1.csv* and *test2.csv*.

For all the three datasets - *train.csv*, *test1.csv* and *test2.csv*, print the:

1. Model summary (only for *train.csv*),
2. Confusion matrices,
3. Overall classification accuracies, and
4. Recall

Note that:

1. You cannot use `duration` as a predictor because its value is determined after the marketing call ends. However, after the call ends, we already know whether the client responded positively or negatively. That is why we have used `duration` only for inference in the previous questions. It helped us understand the effect of the length of the call on marketing success.
2. It is possible to develop the model satisfying constrains (a) and (b) with just appropriate transformation(s) of the predictor(s). However, you may consider interactions if you wish. Justify the transformations, if any, with visualizations.
3. You are free to choose any value of the decision threshold probability for classifying observations. However, you must use the same threshold on all the three datasets.

(15 points)

D.14 ROC-AUC

Report the probability that the model will predict a higher probability of response for a customer who signs up for the term deposit as compared to the customer who does not sign up, i.e., the ROC-AUC of the developed model in D.13.

Hint: Use the functions `roc_curve`, and `auc` from the `sklearn.metrics` module

(3 points)

D.15 Net-profit

Suppose that the model developed in D.13 is used to predict the clients in *test1.csv* and *test2.csv* who will respond positively to the campaign. Only those clients who are predicted to respond positively are called during the marketing campaign. Assume that:

1. A profit of \$100 is associated with a client who responds positively to the campaign,
2. A loss of \$10 is associated with a client who responds negatively to the campaign

What is the net profit from the campaign? Use the confusion matrices printed in D.13.

(4 points)

D.16 Decision threshold probability

Based on the profit and loss associated with client responses specified in D.15, and the model developed in D.13, find the decision threshold probability of classification, such that the net profit is maximized. Use *train.csv*

Proceed as follows:

1. You would have obtained FPR and TPR for all potential decision threshold probabilities in D.14.
2. Formulate an expression quantifying the net profit per client, in terms of FPR, TPR, and the overall response rate, i.e., proportion of people actually subscribing to the term deposit.
3. Find the decision threshold probability that maximizes the expression in (2).

(5 points)

D.17 Net profit based on new decision threshold probability

Using the new decision threshold probability obtained in D.16, answer D.15, i.e., what is the net-profit associated with the clients in *test1.csv* and *test2.csv* if a marketing campaign is performed. Again, only those clients who are predicted to respond positively, based on the new decision threshold probability, are called during the marketing campaign

Also, print the confusion matrices for predictions on *test1.csv* and *test2.csv* with the new threshold probability.

(4 points)

D.18 Model preference

Was the classification accuracy of the model in D.13 higher than that of the model in D.17? If yes, then should you prefer the model in D.13 for the marketing campaign? Why or why not?

Note: The model in D.17 is the same as in D.13, except with a different decision threshold probability

(3 points)

D.19 ROC curve

Plot the ROC curve for the model developed in D.13. Mark the point on the curve corresponding to the decision threshold probability identified in D.16.

Note that the ROC curve is independent of the decision threshold probability used by the model for prediction

(3 points)

D.20 Profit with TPR / FPR

Make a scatterplot of TPR vs FPR, and color the points based on net profit per client.

You can use the following code to make the plot if you have the relevant metrics in `tpr`, `fpr`, and `net_profit`

(1 point)

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(font_scale=1.5)
plt.rcParams["figure.figsize"] = (9,6)
plt.rcParams["figure.autolayout"] = True
f, ax = plt.subplots()
points = ax.scatter(fpr, tpr, c = net_profit, s=50, cmap="Blues")
f.colorbar(points, label = "Net profit ($) \n(per client)")
plt.xlabel("False positive rate")
plt.ylabel("True positive rate")
plt.show()
```

D.21 Precision-recall

Compare the precision and recall of the models in D.13 and D.17 on *train.csv*.

Note: The model in D.17 is the same as in D.13, except with a different decision threshold probability

(4 points)

D.22 Precision-recall: important metric

Based on the above comparison, which metric among precision and recall turns out to be more important for maximizing the net profit in the marketing campaign?

(1 point)

D.23 Precision-recall curve

Plot the precision-recall curve vs decision threshold probability for the model developed in D.13. Mark the points on the curve corresponding to the decision threshold probability identified in D.16.

(3 points)

D.24 Precision-recall vs FPR-TPR

Instead of using the FPR and TPR metrics to find the optimum decision threshold probability in D.16, use the precision-recall metrics to find the same.

(5 points)

E Assignment E

1. You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and not as a group activity.
2. Do not write your name on the assignment.
3. Write your code in the *Code* cells and your answer in the *Markdown* cells of the Jupyter notebook. Ensure that the solution is written neatly enough to understand and grade.
4. Use Quarto to print the *.ipynb* file as HTML. You will need to open the command prompt, navigate to the directory containing the file, and use the command: `quarto render filename.ipynb --to html`. Submit the HTML file.
5. The assignment is worth 100 points, and is due on **Tuesday, 7th March 2023 at 11:59 pm**.
6. **Five points are properly formatting the assignment**. The breakdown is as follows:

- Must be an HTML file rendered using Quarto (1 pt). *If you have a Quarto issue, you must mention the issue & quote the error you get when rendering using Quarto in the comments section of Canvas, and submit the ipynb file.*
- No name can be written on the assignment, nor can there be any indicator of the student's identity—e.g. printouts of the working directory should not be included in the final submission (1 pt)
- There aren't excessively long outputs of extraneous information (e.g. no printouts of entire data frames without good reason, there aren't long printouts of which iteration a loop is on, there aren't long sections of commented-out code, etc.) (1 pt)
- Final answers of each question are written in Markdown cells (1 pt).
- There is no piece of unnecessary / redundant code, and no unnecessary / redundant text (1 pt)

Calculating Root Mean Square Error (RMSE) in Sklearn

You may use sklearn to compute the RMSE.

`sklearn.metrics` has `mean_squared_error` function with a `squared` kwarg (default to `True`). Setting `squared` to `False` will return the RMSE

```
from sklearn.metrics import mean_squared_error
rmse = mean_squared_error(y_actual, y_predicted, squared=False)
```

E.1 Energy model

The datasets *ENB2012_Train.csv* and *ENB2012_Test.csv* provide data on energy analysis using 12 different building shapes simulated in Ecotect. The buildings differ with respect to the glazing area, the glazing area distribution, and the orientation, amongst other parameters. Below is the description of the data columns:

1. `X1`: Relative Compactness
2. `X2`: Surface Area
3. `X3`: Wall Area
4. `X4`: Roof Area
5. `X5`: Overall Height
6. `X6`: Orientation
7. `X7`: Glazing Area
8. `X8`: Glazing Area Distribution
9. `Y1`: Heating Load

E.1.1

Suppose that we want to implement the best subset selection algorithm to find the first order predictors (`X1-X8`) that can predict heating load (`Y1`). How many models for $E($`Y1`$)$ are possible, if the model includes (i) one variable, (ii) three variables, and (iii) eight variables? Show your steps without running any code.

Note: The notation $E($`Y1`$)$ means the expected value of `Y1` or the mean of `Y1`.

(3 points)

E.1.2

Implement the best subset selection algorithm to find the *best* first-order predictors of heating load `Y1`. Print out the model summary.

Note:

1. Use *ENB2012_Train.csv* and consider only the first-order terms.
2. Use the BIC criterion for model selection.

(4 points)

E.1.3

Should R-squared be used to select from among a set of models with different numbers of predictors? Justify your answer.

(1 point for answer, 2 points for justification)

E.1.4

Calculate the RMSE of the model found in E.2. Compare it with the RMSE of the model using all first-order predictors. You will find that the two RMSEs are similar. Seems like the best subset model didn't help improve prediction.

1. Why did the best subset model not help improve prediction accuracy as compared to the model with all the predictors?
2. Does the best subset model provide a more accurate inference as compared to the model with all the predictors? Why or why not?

Hint: **V**ery **I**mportant **F**act!

(2 points for computing the RMSEs, 3 + 3 points for justifications)

E.1.5

Let us consider adding all the 2-factor interactions of the predictors in the model. Answer the following questions without running code.

1. How many predictors do we have in total?
2. Assume best subset selection is used. How many models are fitted in total?
3. Assume forward selection is used. How many models are fitted in total?
4. Assume backward selection is used. How many models are fitted in total?
5. How many models will be developed in the iteration that contains exactly 10 predictors in each model – for best subsets, fwd/bwd regression?
6. What approach would you choose for variable selection (amongst best subset, forward selection, backward selection)?

(6 x 2 = 12 points)

E.1.6

Use forward selection to find the *best* first-order predictors and 2-factor interactions of the predictors of `Y1` (Heating Load). Print out the model summary.

(5 points)

E.1.7

Calculate the RMSE of the model found in E.1.6. Compare it with:

1. the RMSE of model you found in E.1.2 and,
2. the RMSE of the model using all the predictors and all their 2-factor interaction terms.

Among the three models (the model developed in E.1.2, the model developed in E.1.6, the model that has all the predictors and all their 2-factor interactions), discuss which model will you prefer for prediction, and which model will you prefer for inference, and why?

(2 points for computing the RMSEs, 3 + 3 points for discussion)

E.1.8

Assume that we found another dataset of 32 variables on the same set of 768 buildings (542 for training) that we would want to add into our model. We want find the "best" model of all 40 predictors and their 2-factor interaction terms. Would you choose forward or backward selection? Justify your answer.

(1 point for answer, 4 points for justification)

E.2 Planetary radius model

See https://exoplanetarchive.ipac.caltech.edu (for context/source). We are using the Composite Planetary Systems dataset

E.2.1

Say we're interested in modeling the radius of exoplanets in kilometers, which is named as `pl_rade` in the data. Note that the variable `pl_rade` captures the radius of each planet as a proportion of Earth's radius, which is approximately 6,378.1370 km.

Develop a linear regression model to predict `pl_rade` using all the variables in *train_CompositePlanetarySystems.* except `pl_name`, `disc_facility` and `disc_locale`. Find the RMSE (Root mean squared error) of the model on *test1_CompositePlanetarySystems.csv* and *test2_CompositePlanetarySystems.csv*.

(4 points)

E.2.2

Develop a ridge regression model to predict `pl_rade` using all the variables in *train_CompositePlanetarySystems.* except `pl_name`, `disc_facility` and `disc_locale`. What is the optimal value of the tuning parameter λ?

Hint: You may use the following grid of lambda values to find the optimal λ: `alphas = 10**np.linspace(2,0.5,200)*0.5`

Remember to standardize data before fitting the ridge regression model

(5 points)

E.2.3

Use the optimal value of λ found in the previous question to develop a ridge regression model. What is the RMSE of the model on *test1_CompositePlanetarySystems.csv* and *test2_CompositePlanetarySystems.csv*?

(5 points)

E.2.4

Note that ridge regression has a much lower RMSE on test datasets as compared to Ordinary least squares (OLS) regression. Shrinking the coefficients has reduced the variance of the estimated coefficients with a little increase in bias, thereby improving the model fit. Appreciate it. Which are the top two predictors for which the coefficients have shrunk the most?

To answer this question, find the ridge regression estimates for $\lambda = 10^{-10}$ *(almost zero regularization)*. Treat these estimates as OLS estimates and find the predictors for which these estimates have shrunk the most as compared to the model developed in E.2.3.

(4 points for code, 1 point for answer)

E.2.5

Why do you think the coefficients of the two variables identified in the previous question shrunk the most?

(4 points for justification - including code used)

E.2.6

Develop a lasso model to predict `pl_rade` using all the variables in *train_CompositePlanetarySystems.csv* except `pl_name`, `disc_facility` and `disc_locale`. What is the optimal value of the tuning parameter λ?

Hint: You may use the following grid of lambda values to find the optimal λ: `alphas = 10**np.linspace(0,-2.5,200)*0.5`

(4 points)

E.2.7

Use the optimal value of λ found in the previous question to develop a lasso model. What is the RMSE of the model on *test1_CompositePlanetarySystems.csv* and *test2_CompositePlanetarySystems.csv*?

(5 points)

E.2.8

Note that lasso has a much lower RMSE on test datasets as compared to Ordinary least squares (OLS) regression. Shrinking the coefficients has improved the model fit. Appreciate it. Which variables have been eliminated by lasso?

To answer this question, find the predictors whose coefficients are 0 in the lasso model.

(2 points for code, 1 point for answer)

E.3 K-fold cross validation

We haves used *car_features_train.csv* and *car_prices_train.csv* in class notes to predict car `price`. Based on correlation with `price`, the four most relevant continuous predictors to predict car price are `year`, `mpg`, `mileage`, and `engineSize`. In this question, you will use K-fold cross validation to find out the relevant interactions of these predictors and the relevant interactions of the polynomial transformations of these predictors for predicting car `price`. We'll consider quadratic and cubic transformations of each predictor, and the interactions of these transformed predictors. For example, some of the interaction terms that you will consider are (`year`^{2}), (`year`)(`mpg`), (`year`^{2})(`mpg`), (`year`)(`mpg`)(`mileage`), (`year`)(`mpg`^{2})(`mileage`), (`year`)(`mpg`^{2})(`mileage`)(`engineSize`^{3}), etc. The highest degree interaction term will be of degree 12 - (`year`^{3})(`mpg`^{3})(`mileage`^{3})(`engineSize`^{3}), and the lowest degree interaction terms will be of degree two, such as (`engineSize`^{2}), (`engineSize`)(`mpg`), etc.

The algorithm to find out the relevant interactions using K-fold cross validation is as follows. Most of the algorithm is already coded for you. You need to code only step 2 as indicated below.

1. Start with considering interactions of degree `d` = 2:
2. **Find out the 5-fold cross validation RMSE if an interaction of degree `d` is added to the model** *(You need to code only this step).*
3. Repeat step (2) for all possible interactions of degree `d`.
4. Include the interaction of degree `d` in the model that leads to the highest reduction in the 5-fold cross validation error as compared to the previous model *(forward stepwise selection based on K-fold cross validation)*
5. Repeat steps 2-4 until no more reduction is possible in the 5-fold cross validation RMSE.
6. If `d` = 12, then stop, otherwise increment `d` by one, and repeat steps 2-5.

The above algorithm is coded below. The algorithm calls a function `KFoldCV` to compute the 5-fold cross validation RMSE given the interaction terms already selected in the model are `selected_interactions`, and the interaction term to be tested is `interaction_being_tested`. The function must return the 5-fold cross validation RMSE if the `interaction_being_tested` is included to the model consisting of `year`, `mpg`, `mileage`, and `engineSize` as predictors in addition to the already added interactions in `selected_interactions`. The model equation for which you need to find the 5-fold cross validation RMSE will be`'price~year+mpg+mileage+engineSize' + selected_interactions + interaction_being_tested`

You need to do the following:

1. Fill out the function `KFoldCV`.

2. Execute the code to obtain the relevant interactions in `selected_interactions`. Print out the object `selected_interactions`.
3. Fit the model with the four predictors, the `selected_interactions` and compute the RMSE on test data.

Relevance of this question to the linear regression prediction problem: Once you figure out the four most useful predictors to predict `money_made_inv`, use this algorithm to find out their useful interactions. Combine the model with the EDA-based insight of developing the model only where `out_prncp_inv`>0, and you should get a RMSE of less than 400 on the public leaderboard! *(This algorithm is inspired by Victoria Shi's solution to the linear regression prediction problem).*

Note that this brute-force approach of finding relevant interactions may work sometimes, especially when $n >> p$ *(the number of observations are much higher than the number of predictors)*, and/or if the relationship can approximated as a linear combination of polynomial interactions. However, it is unlikely to work in case of relatively less number of observations, and in case of highly non-linear relationships that are difficult to approximate with polynomial transformations. In such problems, a few minutes spent on EDA to figure out transformations, etc. may help develop a better model than this brute-force approach. For example, EDA-based transformations helped you get a RMSE of less than $350k in question C.1.6, while this approach gives a relatively much higher RMSE. Note that the data in C.1.6 has a relatively less number of observations.

(10 (filling out the function) + 1 + 1 points)

```
import pandas as pd
import numpy as np
trainf = pd.read_csv('Car_features_train.csv')
trainp = pd.read_csv('Car_prices_train.csv')
testf = pd.read_csv('Car_features_test.csv')
testp = pd.read_csv('Car_prices_test.csv')
test = pd.merge(testf, testp)
train = pd.merge(trainf, trainp)
```

```
# Creating a dataframe that will consist of all combinations of polynomial transformations
# predictors to be considered for interactions

predictor_set = ['year','mpg','engineSize','mileage']
from itertools import product

#Considering quadratic and cubic transformations
values = np.arange(0,4)
```

```
polynomial_transformations = pd.DataFrame(product(values, repeat=4), columns=predictor_set
polynomial_transformations.loc[:,'sum_degree'] = (polynomial_transformations).astype(int).
polynomial_transformations.loc[:,'count_zeros'] = (polynomial_transformations == 0).astype
polynomial_transformations.sort_values(by = ['count_zeros', 'sum_degree'], ascending=[Fals
polynomial_transformations.drop(columns = ['count_zeros'], inplace=True)
polynomial_transformations.reset_index(inplace = True, drop = True)
```

```
#Setting the seed as we are shuffling the data before splitting it into K-folds
np.random.seed(123)
# Shuffling the training set before creating K folds
train = train.sample(frac=1)
k = 5 #5-fold cross validation
fold_size = np.round(train.shape[0]/k)
```

```
# Fill out this function - that is all you need to do to make the code work!

# The function must return the mean 5-fold cross validation RMSE for the model
# that has the 4 individual predictors - 'year', 'mpg', 'mileage', and 'engineSize',
# the 'selected_interactions', and the 'interaction_being_tested'

# Uncomment the lines below and fill the function
#def KFoldCV(selected_interactions, interaction_being_tested):

        # model = sm.ols('price~year+mpg+mileage+engineSize'+selected_interactions+\
        # interaction_being_tested, data = ...).fit()

    #return
```

```
# This code implements the algorithm of systematically considering interactions of degree
# the interaction of degree 12. For a given degree 'd' the interactions are selected greed
# highest reduction in the 5-fold cross validation RMSE. Once no more reduction in the 5-f
# RMSE is possible using interactions of degree 'd', interaction terms of the next higher

# 5-fold cross validation RMSE of the initial model with the 4 predictors of degree one
cv_previous_model = KFoldCV(selected_interactions = '', interaction_being_tested = '')
interaction_being_tested = '+'
selected_interactions = ''

# Considering interactions of degree 'd' = 2 to 12
for d in np.arange(2,13):
```

```
# Selecting interaction terms of degree = 'd'
degree_set = polynomial_transformations.loc[polynomial_transformations.sum_degree==d,

# Initializing objects to store the interactions of degree 'd' that reduce the
# 5-fold cross validation RMSEs as compared to the previous model
interactions_that_reduce_KfoldCV = []; cv_degree = [];

# Creating another DataFrame that will consist of the updated set of interactions of d
# as interactions that do not reduce the 5-fold cross validation RMSE will be discarde
degree_set_updated = pd.DataFrame(columns = degree_set.columns)

# Continue adding interactions of degree 'd' in the model until no interactions reduce
# the 5-fold cross-validation RMSE
while True:

    #Iterating over all possible interactions of degree 'd'
    for index, row in degree_set.iterrows():

        # Creating the formula expression for the interaction term to be tested
        for predictor in predictor_set:
            interaction_being_tested = interaction_being_tested + ('I('+predictor +'**
                                    str(row[predictor]) + ')*' if row[predictor]>1 el
                                          predictor + '*' if row[predictor]==1 else '
        interaction_being_tested = interaction_being_tested[:-1]

        # Call the function 'KFoldCV' to find out the 5-fold cross validation error on
        # interaction term being tested to the model
        cv = KFoldCV(selected_interactions, interaction_being_tested)

        # If the interaction term being tested reduces the 5-fold cross validation RMS
        # previous model, then consider adding it to the model
        if cv<cv_previous_model:
            interactions_that_reduce_KfoldCV.append(interaction_being_tested)
            cv_degree.append(cv)
            degree_set_updated = pd.concat([degree_set_updated, row.to_frame().T])
        interaction_being_tested = '+'
    cv_data = pd.DataFrame({'interaction':interactions_that_reduce_KfoldCV, 'cv':cv_de

    # Sort the interaction terms that reduce the 5-fold cross valdiation RMSE based on
    # 5-fold cross validation RMSE
    cv_data.sort_values(by = 'cv', inplace = True)
```

```
# Break the loop if no interaction of degree 'd' reduces the 5-fold cross validati
# compared to the previous model
if cv_data.shape[0]==0:
    break

# Select the interaction that corresponds to the least 5-fold cross validation RMS
selected_interactions = selected_interactions + cv_data.iloc[0,0]
cv_previous_model = cv_data.iloc[0,1]
cv_degree = []; interactions_that_reduce_KfoldCV = []
degree_set = degree_set_updated.copy()
degree_set_updated = pd.DataFrame(columns = degree_set.columns)

# Print the progress after each model update, i.e., after an interaction term is s
print("Degree of interactions being considered:",d, ", 5-fold CV RMSE:", cv_previo
```

F Assignment E (Section 22)

1. You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and not as a group activity.
2. Do not write your name on the assignment.
3. Write your code in the *Code* cells and your answer in the *Markdown* cells of the Jupyter notebook. Ensure that the solution is written neatly enough to understand and grade.
4. Use Quarto to print the *.ipynb* file as HTML. You will need to open the command prompt, navigate to the directory containing the file, and use the command: `quarto render filename.ipynb --to html`. Submit the HTML file.
5. The assignment is worth 100 points, and is due on **Tuesday, 7th March 2023 at 11:59 pm**.
6. **Five points are properly formatting the assignment**. The breakdown is as follows:

- Must be an HTML file rendered using Quarto (1 pt). *If you have a Quarto issue, you must mention the issue & quote the error you get when rendering using Quarto in the comments section of Canvas, and submit the ipynb file.*
- No name can be written on the assignment, nor can there be any indicator of the student's identity—e.g. printouts of the working directory should not be included in the final submission (1 pt)
- There aren't excessively long outputs of extraneous information (e.g. no printouts of entire data frames without good reason, there aren't long printouts of which iteration a loop is on, there aren't long sections of commented-out code, etc.) (1 pt)
- Final answers of each question are written in Markdown cells (1 pt).
- There is no piece of unnecessary / redundant code, and no unnecessary / redundant text (1 pt)

Calculating Root Mean Square Error (RMSE) in Sklearn

`sklearn.metrics` has `mean_squared_error` function with a `squared` kwarg (default to `True`). Setting `squared` to `False` will return the RMSE

```
from sklearn.metrics import mean_squared_error
rmse = mean_squared_error(y_actual, y_predicted, squared=False)
```

Energy model

The datasets *ENB2012_Train.csv* and *ENB2012_Test.csv* provide data on energy analysis using 12 different building shapes simulated in Ecotect. The buildings differ with respect to the glazing area, the glazing area distribution, and the orientation, amongst other parameters. Below is the description of the data columns:

1. `X1`: Relative Compactness
2. `X2`: Surface Area
3. `X3`: Wall Area
4. `X4`: Roof Area
5. `X5`: Overall Height
6. `X6`: Orientation
7. `X7`: Glazing Area
8. `X8`: Glazing Area Distribution
9. `y1`: Heating Load

F.1 E.1.1

Suppose that we want to implement the best subset selection algorithm to find the first order predictors (`X1-X8`) that can predict heating load (`y1`). How many models for $E(\texttt{y1})$ are possible, if the model includes (i) one variable, (ii) three variables, and (iii) eight variables? Show your steps without running any code.

Note: The notation $E(\texttt{y1})$ means the expected value of `y1` or the mean of `y1`.

(3 points)

F.2 E.1.2

Implement the best subset selection algorithm to find the *best* first-order predictors of heating load `y1`.

Note:

1. Use *ENB2012_Train.csv* and consider only the first-order terms.

2. Use the BIC criterion for model selection.

(4 points)

F.3 E.1.3

Should R-squared be used to select from among a set of models with different numbers of predictors? Justify your answer.

(1 point for answer, 2 points for justification)

F.4 E.1.4

Calculate the RMSE of the model found in E.2. Compare it with the RMSE of the model using all first-order predictors. You will find that the two RMSEs are similar. Seems like the best subset model didn't help improve prediction.

1. Why did the best subset model not help improve prediction accuracy as compared to the model with all the predictors?
2. Does the best subset model provide a more accurate inference as compared to the model with all the predictors? Why or why npt?

Hint: **V**ery **I**mportant **F**act!

(2 points for computing the RMSEs, 3 + 3 points for justifications)

F.5 E.1.5

Let us consider adding all the 2-factor interactions of the predictors in the model. Answer the following questions without running code.

1. How many predictors do we have in total?
2. Assume best subset selection is used. How many models are fitted in total?
3. Assume forward selection is used. How many models are fitted in total?
4. Assume backward selection is used. How many models are fitted in total?
5. How many models will be developed in the iteration that contains exactly 10 predictors in each model – for best subsets, fwd/bwd regression?

6. With sklearn.feature_selection.SequentialFeatureSelector, how many models will be deleloped when setting the `n_features_to_select` to 10 for forward selection and backward selection respectively?

*(6*2 = 12 points)**

F.6 E.1.6

Use forward selection to find the *best* first-order predictors and 2-factor interactions of the predictors of `y1` (Heating Load).

(5 points)

F.7 E.1.7

Use forward selection in sklearn.feature_selection.SequentialFeatureSelector to find the *best* first-order predictors and 2-factor interactions of the predictors of `y1` (Heating Load), setting the `n_features_to_select` to the number of predictors of the best model found in E.1.6.

Is the best subset found using sklearn the same as the one found in E.1.5. why or why not?

(5 points)

F.8 E.1.8

Calculate the RMSE of the model found in E.1.7. Compare it with:

1. the RMSE of model you found in E.1.2 and,
2. the RMSE of the model using all the predictors and all their 2-factor interaction terms.

Among the 4 models (the model developed in E.1.2, the model developed in E.1.7, the model that has all the predictors and all their 2-factor interactions), discuss which model will you prefer for prediction, and which model will you prefer for inference, and why?

(2 points for computing the RMSEs, 3 + 3 points for discussion)

F.9 E.1.9

Assume that we found another dataset of 32 variabels on the same set of 768 buildings (542 for training) that we would want to add into our model. We want find the "best" model of all 40 predictors and their 2-factor interaction terms. Would you choose forward or backward selection? Justify your answer.

(1 point for answer, 4 points for justification)

Planetary radius model

See https://exoplanetarchive.ipac.caltech.edu (for context/source). We are using the Composite Planetary Systems dataset

F.10 E.2.1

Say we're interested in modeling the radius of exoplanets in kilometers, which is named as `pl_rade` in the data. Note that the variable `pl_rade` captures the radius of each planet as a proportion of Earth's radius, which is approximately 6,378.1370 km.

Develop a linear regression model to predict `pl_rade` using all the variables in *train_CompositePlanetarySystems.* except `pl_name`, `disc_facility` and `disc_locale`. Find the RMSE (Root mean squared error) of the model on *test1_CompositePlanetarySystems.csv* and *test2_CompositePlanetarySystems.csv*.

(4 points)

F.11 E.2.2

Develop a ridge regression model to predict `pl_rade` using all the variables in *train_CompositePlanetarySystems.* except `pl_name`, `disc_facility` and `disc_locale`. What is the optimal value of the tuning parameter λ?

Hint: You may use the following grid of lambda values to find the optimal λ: `alphas = 10**np.linspace(2,0.5,200)*0.5`

Remember to standardize data before fitting the ridge regression model

(5 points)

F.12 E.2.3

Use the optimal value of λ found in the previous question to develop a ridge regression model. What is the RMSE of the model on *test1_CompositePlanetarySystems.csv* and *test2_CompositePlanetarySystems.csv*?

(5 points)

F.13 E.2.4

Note that ridge regression has a much lower RMSE on test datasets as compared to Ordinary least squares (OLS) regression. Shrinking the coefficients has reduced the variance of the estimated coefficents with a little increase in bias, thereby improving the model fit. Appreciate it. Which are the top two predictors for which the coefficients have shrunk the most?

To answer this question, find the ridge regression estimates for $\lambda = 10^{-10}$ *(almost zero regularization)*. Treat these estimates as OLS estimates and find the predictors for which these estimates have shrunk the most as compared to the model developed in E.2.3.

(4 points for code, 1 point for answer)

F.14 E.2.5

Why do you think the coefficients of the two variables identified in the previous question shrunk the most?

(4 points for justification - including code used)

F.15 E.2.6

Develop a lasso model to predict `pl_rade` using all the variables in *train_CompositePlanetarySystems.csv* except `pl_name`, `disc_facility` and `disc_locale`. What is the optimal value of the tuning parameter λ?

Hint: You may use the following grid of lambda values to find the optimal λ: `alphas = 10**np.linspace(0,-2.5,200)*0.5`

(4 points)

F.16 E.2.7

Use the optimal value of λ found in the previous question to develop a lasso model. What is the RMSE of the model on *test1_CompositePlanetarySystems.csv* and *test2_CompositePlanetarySystems.csv*?

(5 points)

F.17 E.2.8

Note that lasso has a much lower RMSE on test datasets as compared to Ordinary least squares (OLS) regression. Shrinking the coefficients has improved the model fit. Appreciate it. Which variables have been eliminated by lasso?

To answer this question, find the predictors whose coefficients are 0 in the lasso model.

(2 points for code, 1 point for answer)

F.18 E.3

We haves used *car_features_train.csv* and *car_prices_train.csv* in class notes to predict car `price`. Based on correlation with `price`, the four most relevant continuous predictors to predict car price are `year`, `mpg`, `mileage`, and `engineSize`. In this question, you will use K-fold cross validation to find out the relevant interactions of these predictors and the relevant interactions of the polynomial transformations of these predictors for predicting car `price`. We'll consider quadratic and cubic transfromations of each predictor, and the interactions of these transformed predictors. For example, some of the interaction terms that you will consider are (year^2), (year)(mpg), (year^2)(mpg), (year)(mpg)(mileage), (year)(mpg^2)(mileage), (year)(mpg^2)(mileage)(engineSize^3), etc. The highest degree interaction term will be of degree 12 - (year^3)(mpg^3)(mileage^3)(engineSize^3), and the lowest degree interaction terms will be of degree two, such as (engineSize^2) or (engineSize)(mpg), etc.

The algorithm to find out the relevant interactions using K-fold cross validation is as follows. Most of the algorithm is already coded for you. You need to code only part 2 as indicated below.

1. Start with considering interactions of degree `d` = 2:
2. **Find out the 5-fold cross validation RMSE if an interaction of degree `d` is added to the model** *(You need to code only this part).*
3. Repeat step (2) for all possible interactions of degree `d`.

4. Include the interaction of degree `d` in the model that leads to the highest reduction in the 5-fold cross validation error as compared to the previous model *(forward stepwise selection based on K-fold cross validation)*
5. Repeat steps 2-4 until no more reduction is possible in the 5-fold cross validation RMSE.
6. If `d` = 12, then stop, otherwise increment `d` by one, and repeat steps 2-5.

The above algorithm is coded below. The algorithm calls a function `KFoldCV` to compute the 5-fold cross validation RMSE given the interaction terms already selected in the model as `selected_interactions`, and the interaction term to be tested as `interaction_being_tested`. The function must return the 5-fold cross validation RMSE if `interaction_being_tested` is included to the model consisting of `year`, `mpg`, `mileage`, and `engineSize` in addition to the already added interactions in `selected_interactions`. The features for which you need to find the 5-fold cross validation RMSE will be `year+mpg+mileage+engineSize'+selected_interactions+interaction_being_tested`

You need to do the following:

1. Fill out the function `KFoldCV`.
2. Execute the code to obtain the relevant interactions in `selected_interactions`. Print out the object `selected_interactions`.
3. Fit the model with the four predictors, the `selected_interactions` and compute the RMSE on test data.

Relevance of this question to the linear regression prediction problem: Once you figure out the four most useful predictors to predict `money_made_inv`, use this algorithm to find out their useful interactions. Combine the model with the EDA-based insight of developing the model only where `out_prncp_inv`>0, and you should get a RMSE of less than 400 on the public leaderboard! *(This algorithm is inspired by Victoria Shi's solution to the linear regression prediction problem).*

Note that this brute-force approach of finding relevant interactions may work sometimes, especially when $n >> p$ *(the number of observations are much higher than the number of predictors)*. However, in certain problems, a few minutes spent on EDA to figure out transformations, etc. will help develop a better model than this brute force approach. For example, EDA-based transformations helped you get a RMSE of less than $350k on question C.1.6, which is not possible with this approach.

(10 (filling out the function) + 1 + 1 points)

```
import pandas as pd
import numpy as np
from tqdm import tqdm
```

```
#build the train dataset
predictor_set = ['year','mpg','engineSize','mileage']
trainf = pd.read_csv('./Datasets/Car_features_train.csv',index_col=0)
trainp = pd.read_csv('./Datasets/Car_prices_train.csv',index_col=0)
from sklearn.preprocessing import PolynomialFeatures
def make_dataset(features_df, target_df, max_degree=3):
  poly = PolynomialFeatures(max_degree, include_bias=False)
  X=poly.fit_transform(features_df[predictor_set].values)
  df = pd.DataFrame(X, columns=poly.get_feature_names_out(predictor_set), index=features_d
  df['price']=target_df
  df=df.sample(frac=1)
  return df
train=make_dataset(trainf,trainp, 12)
```

```
# Fill out this function - that is all you need to do to make the code work!

# The function must return the mean 5-fold cross validation RMSE for the model
# that has the 'selected_interactions', and the 'interaction_being_tested'
def KFoldCV(selected_interactions, interaction_being_tested=None):
  """
  All the variable names can be found in `train` dataframe.
  selected_interactions: List[String] ->  the list of variable names that current selected
  interaction_being_tested: String    ->  a single variable name that are testing.
  return: float -> mean of RSME of 5-folder validation.
  """
  return 0
```

```
# This code implements the algorithm of systematically considering interactions of degree
# the interaction of degree 12
history=[]

selected_interactions = list(predictor_set)
cv_previous_model = KFoldCV(selected_interactions = selected_interactions, interaction_bei
history.append([cv_previous_model, ",".join(selected_interactions)])

print("Initially, RMSE={}, features={}".format(cv_previous_model, selected_interactions))

candidates = [col for col in train.columns if col not in predictor_set +['price']]

for interaction_being_tested in tqdm(candidates):
  cv=KFoldCV(selected_interactions.copy(), interaction_being_tested)
```

```
    if cv<cv_previous_model:
      selected_interactions.append(interaction_being_tested)
      cv_previous_model=cv
      history.append([cv_previous_model, ",".join(selected_interactions)])
```

```
Initially, RMSE=9561.448654675505, features=['year', 'mpg', 'engineSize', 'mileage']

100%|                                          | 1815/1815 [00:40<00:00, 45.37it/s]
```

G Practice Final Solutions

STAT303-2 (Winter2023)

Angelica Wang, Naoki Ito, Yida Hao, Victoria Shi, Nayada Tantichirasakul, Radhika Todi, Ally Bardas, Mingyi Gong, Yuyan Zhang, Annabel Skubisz, Karrine Denisova, Hoda Fakhari, Catherine Erickson, Anjali Patel, Elena Cantu and Arvind Krishna.
03/09/2023

These solutions are composed by students of the course STAT303-2 (Winter 2023).

G.1 Potential problems

Presence of which of the following potential problems in a linear regression model may lead to statistically significant variables appearing insignificant?

A) Multicollinearity

B) Outliers

C) Overfitting

Answer: A and B

Explanation:

A) Multicollinearity:

Recall, the estimated variance of the coefficient β_j, of the j^{th} predictor X_j, can be expressed as:

$$\hat{var}(\hat{\beta}_j) = \frac{(\hat{\sigma})^2}{(n-1)\hat{var}(X_j)} . \frac{1}{1 - R^2_{X_j|X_{-j}}} \tag{1}$$

If the predictor X_j is collinear with other predictors, $R^2_{X_j|X_{-j}}$ will be large, which in turn will inflate $\hat{var}(\hat{\beta}_j)$. In other words, multicollinearity inflates the standard errors of the coefficients

for which the variables are collinear. Since t-statistic is calculated by dividing the estimated coefficient by its standard error, the t-statistics shrinks, and the corresponding p-value increases. Therefore, the hypothesis test loses the power to reject the null hypotheses, and thus statistically significant variables appearing insignificant.

Another way to think about this can be that if some predictors are collinear, it can be difficult to separate out the individual effects of these variables in the response and significant variables may appear insignificant.

B) Outliers

Recall, the estimate of error variance is given by:

$$\hat{\sigma}^2 = \frac{RSS}{n-2},$$

where RSS is the residual sum of squared errors. Outliers result in an increase in RSS, leading to an increase in the estimated error variance $\hat{\sigma}^2$, which in turn inflates $\hat{var}(\hat{\beta}_j)$. The rest of the explanation follows from the previous explanation on multicollinearity.

C) Overfitting

Overfitting shrinks RSS, which in turn shrinks $\hat{\sigma}^2$, thereby shrinking $\hat{var}(\hat{\beta}_j)$. Thus overfitting will act in way opposite to what we observe in (A) and (B).

G.2 Potential problems

Classify a data point as influential / outlier / high leverage in a linear regression model, based on the description.

A) The data point is likely to have a large effect on the model in terms of prediction: **Influential point**

B) The data point has the potential to have a large effect on the model in terms of prediction: **High leverage point**

C) The data point is likely to inflate the model R-squared: **High leverage point that is not influencial**

D) The data point is unlikely to have a large effect on the model in terms of prediction: **outlier**

Explanation:

See the graphics in class presentation on *Chapter3_Outliers_high_leverage_influential_points*. Think of influential points / high leverage points / outliers as a force (proportional to the

residual corresponding to the point) pulling a canteliver beam. Depending on the position from where you pull the cantilever beam, you may move it too much or too little.

A) **Inluential point** (high leverage & outlier): an outlier with the respect to both the predictor and the response. It has a large effect on the regression line. As shown in the graphics, influence is higher for more extreme outliers with same leverage and for points with higher leverage & similar outlying distance.

B) **High leverage point**: Observations with high leverage have an unusual value for the predictor (ie. lie outside the domain of most points). High leverage point has the potential to have a large affect on the regression line. It is cause for concern if the least squares line is heavily affected by just a couple of observations, because any problems with these points may invalidate the entire fit.

C) If you have a **high leverage point that is not influencial**: The variance of the response may increase in the presence of high leverage points, since an unusual set of predictor values may correspond to an unusual response, which may increase the total variation. However, as the point is not inluential, the increase in the unexplained variation *(the squared residual)* will not be proportionate to the increase in total variation. As R^2 is one minus the ratio of unexplained variation to total variation, it is likely to increase.

D) **Outliers:** As shown in the graphics, outliers very small effect on prediction.

G.3 Autocorrelation

A linear regression model was developed to predict the number of passengers taking a flight per month. The data consists of number of passengers flying each month from January 1949 to December 1960. The autocorrelation plot below shows the correlation of the residuals with the lagged residuals of the model. Choose the most appropriate option.

```
<IPython.core.display.Image object>
```

A) The above plot shows the presence of autocorrelation. The 6-month lagged response is the most appropriate lag to be added as a predictor in the model to address autocorrelation

B) The above plot shows the presence of autocorrelation. The 12-month lagged response is the most appropriate lag to be added as a predictor in the model to address autocorrelation

C) The above plot shows the presence of autocorrelation. The 1-month lagged response is the most appropriate lag to be added as a predictor in the model to address autocorrelation

D) The above plot shows the absence of autocorrelation as the plot must have a cyclical pattern in the presence of autocorrelation

E) The above plot shows the absence of autocorrelation as the one month lagged residual must have the highest correlation with the residual in the presence of autocorrelation

Answer: B

Explanation: As seen in the plot, the residuals are highly correlated (correlation of more than 60%) with lagged residuals of 12 months. This shows the presence of autocorrelation. To address autocorrelation, the 12-month laggged response will be the most appropriate as it has the highest correlation with the response. Thus, it will explain the variation in the respone the most.

There is no need for there to be a cyclical pattern for autocorrelation. Even if one of the lagged residuals are highly correlated with the residual, it shows the presence of autocorrelation.

G.4 Logistic regression (goodness-of-fit)

Which of the following metrics can be used to assess the goodness-of-fit of a logistic regression model?

A) All of these

B) LL-Null

C) Log-Likelihood

D) Df Model

E) R-squared

Answer: Log-Likelihood

Explanation In logistic regression, the response is assumed to follow a Bernoulli distribution, where the probability of success is a function of the predictors and its coefficients *(the model parameters)*. With this assumption, one can compute the the joint probability density of the observed data as a function of the model parameters. This creates a set of probability distributions *(based on different values of model parameters)* that could have generated the data. The algorithm finds the values of the model parameters *(the beta coefficients)* such that the probability of observing the data maximizes. This probability is the likelihood, and its logarithm is the log-likelihood. The higher the log-likelihood, the more probable it is to observe the data. Thus, log-likelihood is a way to measure the goodness-of-fit of the model.

LL-NULL is the log-likelihood of the model with no parameters. This is compared with the log-likelihood of the model with predictors to test if the regression is statistically significant.

Df Model is the number of predictors in the model.

R-squared cannot be used for logistic regression as there are no residuals.

G.5 Logistic regression (threshold probability)

For a logistic regression model, as we increase the decision threshold probability,

A) None of these

B) the recall will reduce or stay the same

C) the ROC-AUC will increase or stay the same

D) the precision will increase or stay the same

E) the classification accuracy will increase or stay the same

Answer: B

Explanation: See class slide on the confusion matrix below.

```
<IPython.core.display.Image object>
```

Increasing threshold probability means that less observations are predicted to be positive. Hence, some TP could turn into FN, reducing the recall. (this might not happen if there is no observations of actual positives between the thresholds). ROC-AUC is independent of the threshold probability. Both precision and classification accuracy might decrease if the number of FP among actual negatives increase more than the increase of TP among actual positives by the shift in the threshold.

G.6 Decision threshold probability

Which of the following metrics is independent of the decision threshold probability?

A) None of these

B) ROC-AUC

C) All of these (except the "None of these" option)

D) Precision

E) Recall

Answer: ROC-AUC

Explanation By changing the threshold, the number of points classified as negative and positive may change, and so TP, FP, TN and FN may change. Recall and precision may change as they are based on these metrics *(TP ,FP, TN, and FN)*. However, the ROC-AUC specifically analyzes different thresholds. The ROC curve is a plot of TPR against FPR for all possible thresholds, and ROC-AUC is the area under the ROC curve, so the value itself is independent from the decision threshold probability.

G.7 Odds

Consider the following logistic regression model:

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

.

Which of the following metrics will depend on the value of x?

A) Odds ratio when x increases by 2 units

B) increase in log odds when x increases by 10 units

C) All of these

D) Increase in predicted probability when x increases by 1 unit

E)none of these

Answer: D

Explanation:

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

$$\implies \log\left(\frac{p(x)}{1 - p(x)}\right) = \beta_0 + \beta_1 x$$

$$\implies \log\left(Odds(x)\right) = \beta_0 + \beta_1 x$$

When x increases by 'c' units,

$$p(x + c) - p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 (x + c))}} - \frac{1}{1 + e^{-(\beta_0 + \beta_1 (x))}}$$

$$log(Odds(x+c)) - log(Odds(x)) = \beta_1 c$$

$$\frac{Odds(x+c))}{Odds(x)} = e^{\beta_1 c}$$

We can see that only the increase in predicted probability when x increases by 1 unit is dependent on x.

G.8 Precision-recall

We develop a logistic regression model to predict whether someone will pay a loan back or not. Loans are "approved" by us only for those borrowers who are predicted to pay back. The positive class is the borrowers that pay back the loans. What would a recall of 81% mean?

A) 81% of the borrowers that would pay back the loan are approved by us: Recall = TP/(TP + FN). TP here are those who are [approved by us] who [pay back the loan], while FN are those who were [not approved by us] but actually [pay back the loans]. The denominator is [all who pay back the loan]. Thus, Recall here means: among [all who pay back the loan], 81% are [approved by us].

B) Of all the loans we approve, 81% pay us back: This is Precision = TP/(TP + FP)

C) Of all the loans we don't approve, 81% would not have paid us back if they were given the loan: This is the proportion of negatives correctly predicted - like precision for the negative class

D) Of all the loans we don't approve, 19% would not have paid us back if they were given the loan: This is the proportion of negatives incorrectly predicted.

Answer: 81% of the borrowers that would pay back the loan are approved by us.

Explanation: Recall = True Positives/(True Positives + False Negatives).

In this case, True positives are those who got approved and would pay back. False Negatives are those we didn't approve, but would pay back. Therefore, 81% Recall means 81% of the borrowers that would pay back the loan are approved by us.

G.9 Variable selection

Which of the following algorithms can be used for variable selection?

A) Lasso

B) Ridge regression

C) Forward stepwise selection

D) Best subset selection

Answer: A,C,D

Explanation: Both lasso and ridge regression are regularized least squares model, where the a shrinkage penalty is added to the ordinary least squares cost function. The shrinkage penalty in ridge regression shrinks the regression coefficients estimate towards zero, but not exactly zero, while the shrinkage penalty in lasso tends to give a set of zero regression coefficients and leads to a sparse model. Therefore, lasso can be used for variable selection, but not ridge regression.

Forward stepwise and best subset selection are variable selection algorithms by fitting multiple models having different combinations/number of predictors and choosing the best model.

G.10 Precision-recall

You are building a facial recognition model to allow people to unlock their phone. If the phone recognizes the person as the authorized user, it will unlock the phone. If it doesn't recognize the user, it will prompt them to try again or try an alternative method (such as a passphrase). The facial recognition model is a classification model that identifies if the person unlocking the phone is the authorized user (positive response) or not (negative response).

Assume that letting a stranger (unauthorized user) unlock the phone is more risky (or more expensive) than not letting the authorized user unlock the phone.

Which of the following metric is the most important to optimize in the model?

A) Precision

B) Classification accuracy

C) Recall

D) ROC-AUC

Answer: Precision

Explanation:

A) Precision: Precision = TP/(TP + FN). Here, FN are those who are falsely assigned as an unauthorized user when they are actually the authorized user. FP are those who are assigned as the authorized user and are actually an unauthorized user. In this case, it's important to optimize precision because it is more important to reduce the number of FP (strangers being recognized as the authorized user) than to reduce the number of FN (authorized user not being recognized).

B) Classification accuracy: This is incorrect because a model with high accuracy but a high FPR would be unacceptable since it would increase the risk of a stranger unlocking the phone.

C) Recall: This is incorrect because a high recall indicates that many of the positive cases are being detected. However, it does not measure the fraction of unauthorized users that the model identifies as authorized. A high FPR could lead to an unauthorized user unlocking the phone, which is a more expensive mistake than an FN.

D) ROC-AUC: This is incorrect because ROC-AUC does not take into account the cost of the positive and negative classes. It only measures how well the model can distinguish between authorized users and unauthorized users.

G.11 Logistic regression

Consider the following logistic regression model:

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}}$$

where assuming the threshold probability for classifying observations is 0.5. All observation with predicted probability greater than 0.5 are classified as belonging to class $y = 1$, while others are classified as belonging to class $y = 0$.

Which of the following plots correctly visualizes the predicted class based on x_1 and x_2?

```
<IPython.core.display.Image object>
```

Answer: D

Explanation:

x_1 will not have an impact on the outcome because its coefficient is 0. When $x_2 > 5, p(x)$ will be less than 0.5 and y will equal 0, as the decisions threshold probability ois 0.5. When $x_2 < 5, p(x)$ will be greater than 0.5 and y will equal 1.

G.12 ROC-AUC

In which of the following cases will ROC-AUC be the most appropriate metric to optimize among the all the performance metrics we have seen in this course.

A) There are wide disparities in the cost of false negatives vs. false positives, for example, predicting if the person has a serious disease.

B) The predicted probabilities will be used to rank observations, instead of classifying them, for example, the Google search engine using the predicted probabilities to rank pages in the decreasing order of relevance to the search query, instead of classifying the observations as 'relevant' and 'not relevant'.

C) We wish to maximize the overall classification accuracy, for example, predicting if a person will vote for the Democrat or the Republican candidate in the US Presidential elections. Here, you may assume that the cost of false positives is similar to the cost of false negatives.

Answer: (B) only

Explanation:

(A) is incorrect because in cases where there are wide disparities in the cost of false negatives vs. false positives, it may be critical to minimize the performance metric associated with a higher loss. For example when predicting if the person has a serious disease, a false positive could lead to expensive and unnecessary medical treatment. Conversely, a false negative could result in a delay in diagnosis and treatment, potentially leading to a worse outcome. Thus, we want to prioritize minimizing false negatives. Since ROC-AUC is decision-threshold invariant, it's not a useful metric for this type of optimization.

(B) is correct. ROC-AUC is scale-invariant. It measures how well predictions are ranked, rather than the absolute values of the predicted probabilities. Check the link.

(C) is incorrect because AUC is classification-threshold-invariant. It measures the quality of the model's predictions irrespective of what classification threshold is chosen. However, the overall accuracy changes with change in decision threshold probability. To maximize overall accuracy, we need to find the optimal decision threshold probability.

G.13 Model selection

Which of the following linear model selection methods can be used when number of predictors is greater than the number of observations in linear regression?

A) Lasso

B) Ridge regression

C) Forward stepwise selection

D) Backward stepwise selection

E) Best subset selection

Answer: A, B, C

Expanation:

When number of predictors is greater than the number of observations, then, in case of ordinary least squares regression, the number of parameters are greater than the number of equations available to estimate those parameters. Thus, there is no unique solution. Also, in equation (1), $R^2_{X_j|X_{-j}}$ is 1, and so $\hat{var}(\widehat{\beta}_j)$ tends to infinity. Thus, it is not possible to fit an ordinary least squares model in this case. As backward stepwise selection begins with a model considering all predictors, it cannot be used as it is not possible to develop a model with all predictors in this case.

The best subset selection must consider models with all possible combination/number of predictors. Howvever, as number of predictors cannot be greater than the number of observations, we cannot develop all possible models, and thus we cannot use best subset selection.

Forward stepwise starts with no predictors and adds one predictor at a time. It is possible to keep adding predictors until the number of predictors *(excluding the intercept)* is one less than the number of observations. From this set of models, the best model can be chosen based on AIC, BIC, or any goodness-of-fit criteria that accounts for the number of predictors. Thus, it is possible to use forward stepwise selection.

The shrinkage penalty in lasso and ridge regression reduces the variance of the coefficents. The variance is infinity without any penalty when the number of predictors is greater than the number of observations. However, as lasso and ridge regression shrink the penalty, it is possible to obtain a unique solution.

The following is just for your information, but beyond the scope of this course: With lasso, there will be at most as many non-zero predictors as the number of observations. With ridge regression, there may be more non-zero predictors as compared to the number of observations.

G.14 Goodness-of-fit

Which of the following metrics can be used to compare the goodness-of-fit of models with different number of predictors?

A) AIC (Akaike Information criterion)

B) R-squared

C) Log-Likelihood

D) Pseudo R-squared

E) LLR p-value

Answer: AIC

Explanation: AIC is correct because it takes into account both the goodness-of-fit of the model and the complexity of the model (number of predictors).

$$AIC = -2logL + 2d,$$

where L is the maximized value of the likelihood function for the estimated model, and d is the number of predictors. From the above equation, we can see that AIC penalizes models with more parameters. Therefore, AIC allows for comparison between models with different numbers of predictors and helps to determine which model is the best fit.

All the other metrics will increase, while the LLR p-value will decrease with increase in number of predictors, and thus cannot be used to compare models with different number of predictors.

G.15 Model selection

Given a set of predictors, which of the following model selection methods guarantees to provide the best 'least squares' linear regression model, based on adjusted R-squared?

A) Best subset selection

B) Forward stepwise selection

C) Backward stepwise selection

D) Linear regression with all the statistically insignificant predictors removed

Answer: A

Explanation: Best subset selection considers every single model possible, while forward and backward selection don't. Therefore, best subset selection necessarily gives the best model, while stepwise selection methods do not. For example, if a model consisting of predictors x_2 and x_3 is the best possible model, with regards to adjusted R-squared, while the model consisting of x_1 is the best one predictor model, then forward stepwise selection will fail to identify the best possible model.

Adjusted R-squared depends on the residuals, among other things. However, the residuals don't relate directly to statistical significance of predictors. Statistical significance of a predictor implies that the predictor is significantly linearly associated with the response, but it does not determine the variation in response explained by the predictor.

G.16 MSE estimate

Which of the following metrics gives the least biased estimate of MSE (mean squared error) on test data?

- Leave-one-out cross validation error
- MSE (mean squared error) on a test dataset (or validation set)
- K-fold cross validation error, where 1<k<n, where n is the number of observations
- All of these

Answer: Leave-one-out cross validation error

Explanation: Leave-one-out cross-validation offers two advantages:

- It provides a much less biased measure of test MSE compared to using a single test set because we repeatedly fit a model to a dataset that contains n-1 observations. - It tends not to overestimate the test MSE compared to using a single test set.

Textbook p200 Details: The test MSE gives us an idea of how well a model will perform on data it hasn't previously seen, i.e. data that wasn't used to "train" the model.

However, the drawback of using only one testing set is that the test MSE can vary greatly depending on which observations were used in the training and testing sets.

One way to avoid this problem is to fit a model several times using a different training and testing set each time, then calculating the test MSE to be the average of all of the test MSE's.

Like the validation set approach, LOOCV involves splitting the set of observations into two parts (test & train). However, instead of creating two subsets of comparable size, LOOCV uses a single observation (x_1, y_1) for the validation set, and the remaining observations $(x_2, y_2), ..., (x_n, y_n)$ for the training set.

The statistical learning method is fit on the $n-1$ training observations, and a prediction $\hat{y_1}$ is made for the excluded observation using its value x_1. Since (x_1, y_1) was not used in the fitting process, $MSE_1 = (y_1 - \hat{y_1})^2$ provides an approximately unbiased estimate for the test error.

G.17 Stepwise with categorical variable

Suppose you have a categorical predictor `gender` in the dataset with 3 distinct values - 'male', 'female', and 'other'. Following are three ways to transform this predictor to make it suitable for forward stepwise selection. Which method is likely to provide the best model and which method is likely to provide the worst model, with regard to prediction accuracy on unknown (test) data?

A) Use the predictor `gender` as it is for forward stepwise selection.

B) Convert the predictor to 3 dummy variables - 'male', 'female', and 'other', where each dummy variable has 0s and 1s, depending on the 'gender', and use the dummy variables for forward stepwise selection, instead of `gender`

C) Replace the values of 'male' to 0, 'female' to 1, and 'other' to 2 in 'gender', and then use `gender` in forward stepwise selection

D) B is likely to provide the best model and, A or C are likely to provide the worst model

E) C is likely to provide the best model and A is likely to provide the worst model

F) C is likely to provide the best model and B is likely to provide the worst model

G) B is likely to provide the best model and A is likely to provide the worst model

H) None of these

Answer: B is likely to provide the best model, and A or C are likely to provide the worst model.

Explanation:

B is likely to provide the best model because breaking down a categorical variable into dummy variables allows us to perform stepwise selection for each class in `gender`. So it gives the algorithm the option to choose among more models than using the predictor `gender` as it is in the selection.

C is likely to provide a worse model as compared to B since it introduces an unreasonable constraint in the model that holding all other predictors constant, the difference in response between *female* and *male* is the same as the difference in response between *other* and *female*. In other words, we are converting categorical variables into ordinals which is constraining the model to find the relationship of the predictor in the order of male-female-other even though that may not be the case.

A is likely to provide a worse model as compared to B because it introuces a constraint in the model to either include all genders, or no gender. If only the *female* gender explains some variation in response, while there is no distinction between *male* and *other*, then the model should not be forced to keep the *male* and *other* categories. Suppose the *other* category

happens to have a very few observations leading to an unstable coefficient *(high standard error)*, then it may reduce the prediction accuracy.

G.18 ROC-AUC

Which of the following statements regarding ROC-AUC (area under the ROC curve) are true, with regard to a binary classification problem where the classes are named as 'positive' and 'negative'?

A) ROC-AUC is zero if the model makes random predictions

B) The larger the ROC-AUC the better the performance of a logistic regression model

C) ROC-AUC is as the probability that the model ranks a random positive observation more highly than a random negative observation, where a higher rank corresponds to a higher predicted probability

D) The ROC-AUC can never be negative

Answer: B, C, D

Explanation:

A) is incorrect because ROC-AUC is 50% if the model makes random predictions.

B) is correct because AUC ranges from 0-1. The larger the ROC-AUC, the better the model is distinguishing between the classes. ROC-AUC is 1.0 when there is perfect seperation of classes based on the predicted probability.

C) is correct because AUC is the probability that a randomly selected positive observation has a higher predicted probability as compared to a randomly selected negative observation.

D) is correct because ROC-AUC is a probability.

G.19 Lasso

The lasso, relative to least squares, is:

A) More flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance

B) Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance

C) Less flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias

D) More flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias

Answer: B

Explanation:

The cost function for lasso consists of the sum of absolute value of coefficients called the shrinkage penalty or the regularization term. This term helps to reduce the complexity of the model by shrinking the regression coefficients towards zero. Because of this penalty, lasso is less flexible *(compared to the least squares)*, as it restricts the range of possible coefficient values, whereas least squares can assign any value to the coefficients. The decreased flexibility means that lasso has less variance in its predictions and more bias.

The penalty terms in the optimization will lead to bias in estimates, leading to less accuracy in prediction. At the same time, reducing the size of coefficients gives them less variance, increasing the accuracy of prediction. Thus, when the increase in bias is less than the decrease in variance, it can lead to improved prediction accuracy. This is a bias-variance trade-off.

G.20 Computational complexity

Arrange the following model selection methods in increasing order of computational complexity:

Ridge regression, best subset selection, forward stepwise selection

Answer: Ridge regression, forward stepwise selection, best subset selection

Explanation: Best subset selection requires the most computation complexity because it considers all 2^p possible models containing subsets of the p predictors. It cannot be used in case of even a slightly large number of predictors. For example, in case of 30 predictors, more than a billion models will need to be developed to find the best subset model.

In forward stepwise selection, the total number of models with p predictors is $\frac{p(p+1)}{2}$. In case of 30 predictors, this will 435 models.

In ridge regression, we need to fit only one model.

G.21 K-fold CV

For optimizing parameters of a model, K-fold cross validation is preferred over the validation set approach (computing error on a test dataset) because:

A) Validation dataset may have observations overlapping with the training dataset

B) Error on validation dataset is likely to be similar to the error on training data, leading to less value addition

C) K-fold cross validation is computationally less expensive

D) Error on the validation dataset can be highly variable

Answer: D

Explanation:

Testing on one validation set is unreliable because the result is highly dependent on the distribution of the data in the test set. By testing on several validation sets, this concern is alleviated to some extent.

K-fold cross validation is computationally more expensive because K-fold requires multiple training/testing iterations in order to generate reliable results. This means that the entire dataset must be split into subsets, trained on each subset, and tested. This process must be repeated K times.

Error on validation dataset is not likely to be similar to the error on training data in scenarios such as overfitting.

The validation dataset approach assigns each observation to either train or test data, but not both.

G.22 Binning

Binning of a continuous predictor, and then using the bins as predictors in logistic regression is likely to be useful when:

A) The proportion of observations belonging to a class have a non-monotonic relationship with the predictor

B) The proportion of observations belonging to a class have a monotonic relationship with the predictor

C) The proportion of observations belonging to a class are almost constant for each bin of the predictor

D) The predictor has low variance:

Answer: A

Explanation: A single coefficient for a predictor will only provide predicted probabilities that are either continuously increasing or continuously decreasing with increasing predictor values. Thus, it will model only a monotonic (non-decreasing / non-increasing) relationship with the

response. In case of a non-monotonic relationship, binning provides the required flexibility to have predicted probabilites that may increase or decrease with increase in predictor values.

If the proportion of observations belonging to a class are almost constant for each bin of the predictor, then the bins do not explain the variation in the response, and thus are not useful.

If the predictor itself is not varying, then it is unlikely to explain the variation in response, and binning cannot help increase the predictor variance.

Coding questions

G.23 Inference - Logistic regression

Develop a logistic regression model to predict if a patient has a risk of a 10 year coronary heart disease. TenYearCHD = 1 means 'Yes' and TenYearCHD = 0 means 'No'. Use all the available predictors plus only one relevant interaction to answer the question below.

Assuming all other predictors are constant, how much percent higher are the odds of male smokers getting diagnosed with heart disease as compared to female smokers. Round up the answer to the nearest integer greater than the answer. For example, if the odds of male smokers getting diagnosed with heart disease are 50.1% higher than that of female smokers, then enter 51 in the box.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score
import statsmodels.formula.api as sm
import itertools
import time
```

```
train = pd.read_csv('./Datasets/train_heart.csv')
test = pd.read_csv('./Datasets/test_heart.csv')
```

```
predictors = list(train.columns)[0:train.shape[1]-1]
```

```
#logistic model with all predictors and interaction of "male"&"currentSmoker"
model = sm.logit('TenYearCHD~male*currentSmoker+' + '+'.join(predictors), data = train).fi
model.summary()
```

Table G.1: Logit Regression Results

Dep. Variable:	TenYearCHD	No. Observations:	2742
Model:	Logit	Df Residuals:	2725
Method:	MLE	Df Model:	16
Date:	Fri, 10 Mar 2023	Pseudo R-squ.:	0.1085
Time:	02:35:34	Log-Likelihood:	-1031.3
converged:	True	LL-Null:	-1156.8
Covariance Type:	nonrobust	LLR p-value:	3.307e-44

```
#Ratio of odds (of having a heart disease) of a male smoker to a female smoker
np.exp(model.params['male']+model.params['male:currentSmoker'])
```

```
1.5992822263126012
```

G.24 Odds

Are the odds of male non-smokers being diagnosed with heart disease even higher than female smokers, based on the model developed in the previous question?

```
#Ratio of odds (of having a heart disease) of a male non-smoker to a female smoker
np.exp(model.params['male']-model.params['currentSmoker'])
```

```
1.5209571795703307
```

G.25 Tuning threshold probability

Among the options below, which is the maximum threshold probability of classifying observations into classes (TenYearCHD = 1 and TenYearCHD = 0), such that the false negative rate less than 20% on both the test and train datasets?

```
#Function to compute confusion matrix and prediction accuracy on training data
def confusion_matrix_train(model,cutoff=0.5):
    # Confusion matrix
    cm_df = pd.DataFrame(model.pred_table(threshold = cutoff))
    #Formatting the confusion matrix
    cm_df.columns = ['Predicted 0', 'Predicted 1']
```

```
    cm_df = cm_df.rename(index={0: 'Actual 0',1: 'Actual 1'})
    cm = np.array(cm_df)
    # Calculate the accuracy
    accuracy = 100*(cm[0,0]+cm[1,1])/cm.sum()
    fnr = 100*cm[1,0]/(cm[1,0]+cm[1,1])
    return cm_df, accuracy, fnr

#Function to compute confusion matrix and prediction accuracy on test data
def confusion_matrix_test(data,actual_values,model,cutoff=0.5):
#Predict the values using the Logit model
    pred_values = model.predict(data)
# Specify the bins
    bins=np.array([0,cutoff,1])
#Confusion matrix
    cm = np.histogram2d(actual_values, pred_values, bins=bins)[0]
    cm_df = pd.DataFrame(cm)
    cm_df.columns = ['Predicted 0','Predicted 1']
    cm_df = cm_df.rename(index={0: 'Actual 0',1:'Actual 1'})
# Calculate the accuracy
    accuracy = 100*(cm[0,0]+cm[1,1])/cm.sum()
    fnr = 100*cm[1,0]/(cm[1,0]+cm[1,1])
# Return the confusion matrix and the accuracy
    return cm_df, accuracy, fnr
```

```
print(confusion_matrix_test(test,test.TenYearCHD,model,0.1))
confusion_matrix_train(model,0.1)
```

```
(          Predicted 0  Predicted 1
Actual 0        388.0        379.0
Actual 1         18.0        129.0, 56.564551422319475, 12.244897959183673)
```

```
(          Predicted 0  Predicted 1
 Actual 0       1091.0       1241.0
 Actual 1         67.0        343.0,
 52.29759299781182,
 16.341463414634145)
```

```
print(confusion_matrix_test(test,test.TenYearCHD,model,0.1))
confusion_matrix_train(model,0.1)
```

```
(          Predicted 0  Predicted 1
Actual 0         388.0        379.0
Actual 1          18.0        129.0, 56.564551422319475, 12.244897959183673)
```

```
(          Predicted 0  Predicted 1
 Actual 0        1091.0       1241.0
 Actual 1          67.0        343.0,
 52.29759299781182,
 16.341463414634145)
```

G.26 Forward stepwise

Use forward stepwise selection to select a logistic regression model for predicting if a patient has a risk of a 10 year coronary heart disease. How many predictors are there in the best model as per the BIC criterion?

```
def best_sub_plots():
    plt.figure(figsize=(20,10))
    plt.rcParams.update({'font.size': 18, 'lines.markersize': 10})

    # Set up a 2x2 grid so we can look at 4 plots at once
    plt.subplot(1, 2, 1)

    # We will now plot a red dot to indicate the model with the largest adjusted R^2 stati
    # The argmax() function can be used to identify the location of the maximum point of a
    plt.plot(models_best["Rsquared"])
    plt.xlabel('# Predictors')
    plt.ylabel('Log likelihood')

    bic = models_best.apply(lambda row: row[1].bic, axis=1)

    plt.subplot(1, 2, 2)
    plt.plot(bic)
    plt.plot(1+bic.argmin(), bic.min(), "or")
    plt.xlabel('# Predictors')
    plt.ylabel('BIC')
```

```
#Function to develop a model based on all predictors in predictor_subset
def processSubset(predictor_subset):
    # Fit model on feature_set and calculate R-squared
```

```
    model = sm.logit('TenYearCHD~' + '+'.join(predictor_subset),data = train).fit(disp=0)
    Rsquared = model.llf
    return {"model":model, "Rsquared":Rsquared}
```

```
#Function to find the best predictor out of p-k predictors and add it to the model contain
def forward(predictors):

    # Pull out predictors we still need to process
    remaining_predictors = [p for p in X.columns if p not in predictors]

    tic = time.time()

    results = []

    for p in remaining_predictors:
        results.append(processSubset(predictors+[p]))

    # Wrap everything up in a nice dataframe
    models = pd.DataFrame(results)

    # Choose the model with the highest RSS
    best_model = models.loc[models['Rsquared'].argmax()]

    toc = time.time()
    print("Processed ", models.shape[0], "models on", len(predictors)+1, "predictors in",

    # Return the best model, along with some other useful information about the model
    return best_model
```

```
def forward_selection():
    models_best = pd.DataFrame(columns=["Rsquared", "model"])

    tic = time.time()
    predictors = []

    for i in range(1,len(X.columns)+1):
        models_best.loc[i] = forward(predictors)
        predictors = list(models_best.loc[i]["model"].params.index[1:])

    toc = time.time()
```

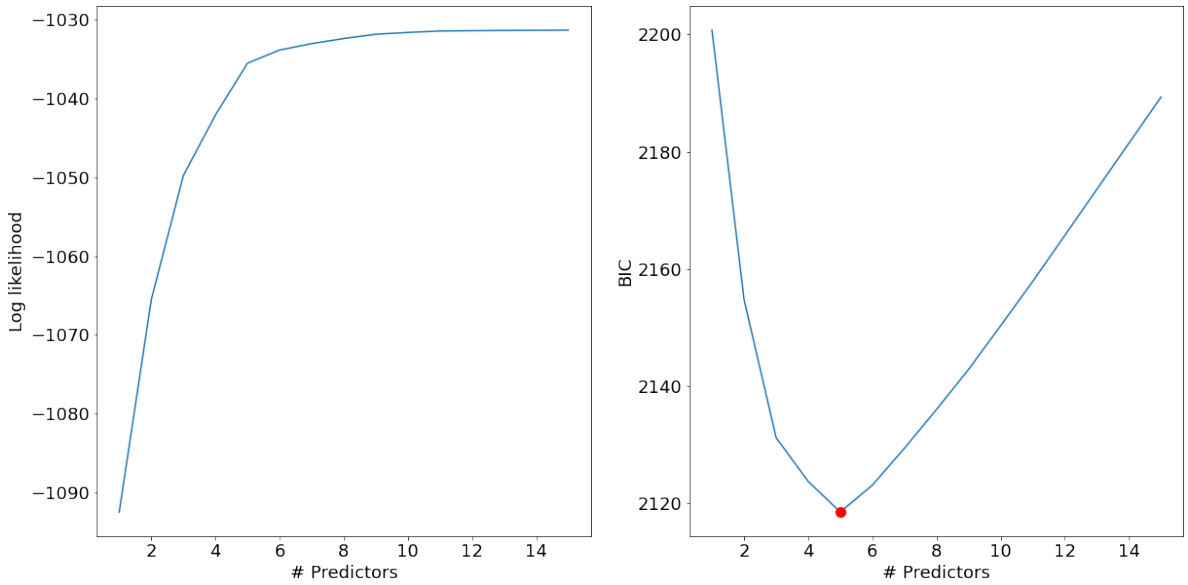
```
    print("Total elapsed time:", (toc-tic), "seconds.")
    return models_best
```

```
X=train.iloc[:,0:train.shape[1]-1]
```

```
models_best = forward_selection()
```

```
Processed  15 models on 1 predictors in 0.10073018074035645 seconds.
Processed  14 models on 2 predictors in 0.08876228332519531 seconds.
Processed  13 models on 3 predictors in 0.09574484825134277 seconds.
Processed  12 models on 4 predictors in 0.10770988464355469 seconds.
Processed  11 models on 5 predictors in 0.1107032299041748 seconds.
Processed  10 models on 6 predictors in 0.10970640182495117 seconds.
Processed  9 models on 7 predictors in 0.10073137283325195 seconds.
Processed  8 models on 8 predictors in 0.10275673866271973 seconds.
Processed  7 models on 9 predictors in 0.09374809265136719 seconds.
Processed  6 models on 10 predictors in 0.14561104774475098 seconds.
Processed  5 models on 11 predictors in 0.08178091049194336 seconds.
Processed  4 models on 12 predictors in 0.06682133674621582 seconds.
Processed  3 models on 13 predictors in 0.07779192924499512 seconds.
Processed  2 models on 14 predictors in 0.04288506507873535 seconds.
Processed  1 models on 15 predictors in 0.020943880081176758 seconds.
Total elapsed time: 1.3882873058319092 seconds.
```

```
best_sub_plots()
```



G.27 Multicollinearity

You are developing a linear regression model to predict 'SalePrice'. Assume all columns except 'Id' and 'SalePrice' to be predictors.

What is the minimum number of predictors to be removed from the model so that there is no multicollinearity?

Assume a VIF of less than 15 indicates absence of multicollinearity.

```
train = pd.read_csv('./Datasets/housing_train.csv')
test = pd.read_csv('./Datasets/housing_test.csv')
```

```
predictors = list(train.columns)[1:train.shape[1]-1]
X = train[predictors]
```

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from statsmodels.tools.tools import add_constant
def vif(X):
    X = add_constant(X)
    vif_data = pd.DataFrame()
    vif_data["feature"] = X.columns
```

```
    for i in range(len(X.columns)):
        vif_data.loc[i,'VIF'] = variance_inflation_factor(X.values, i)

    print(vif_data)
vif(X)
```

```
         feature           VIF
0          const  2.419858e+06
1      MSSubClass  1.480034e+00
2         LotArea  1.342832e+00
3     OverallQual  3.438243e+00
4     OverallCond  1.601301e+00
5       YearBuilt  3.965491e+00
6    YearRemodAdd  2.223979e+00
7      BsmtFinSF1           inf
8      BsmtFinSF2           inf
9       BsmtUnfSF           inf
10    TotalBsmtSF           inf
11     FirstFlrSF           inf
12    SecondFlrSF           inf
13   LowQualFinSF           inf
14      GrLivArea           inf
15   BsmtFullBath  2.242659e+00
16   BsmtHalfBath  1.140692e+00
17       FullBath  2.911473e+00
18       HalfBath  2.190673e+00
19   BedroomAbvGr  2.199007e+00
20   KitchenAbvGr  1.624357e+00
21   TotRmsAbvGrd  4.917930e+00
22     Fireplaces  1.556896e+00
23     GarageCars  5.565460e+00
24     GarageArea  5.397997e+00
25     WoodDeckSF  1.253248e+00
26    OpenPorchSF  1.219440e+00
27  EnclosedPorch  1.242530e+00
28       SsnPorch  1.032740e+00
29    ScreenPorch  1.127507e+00
30       PoolArea  1.095423e+00
31        MiscVal  1.032655e+00
32         MoSold  1.058330e+00
33         YrSold  1.048841e+00
```

```
C:\Users\akl0407\Anaconda3\lib\site-packages\statsmodels\stats\outliers_influence.py:193: Ru
  vif = 1. / (1. - r_squared_i)
```

```
#Remove collinear predictors one at a time
remove = ['BsmtFinSF1']
pred_filter = [x for x in predictors if x not in remove]
X = train[pred_filter]

#Recompute VIF
vif(X)
```

```
         feature           VIF
0          const  2.419858e+06
1      MSSubClass  1.480034e+00
2         LotArea  1.342832e+00
3     OverallQual  3.438243e+00
4     OverallCond  1.601301e+00
5       YearBuilt  3.965491e+00
6    YearRemodAdd  2.223979e+00
7      BsmtFinSF2  1.131439e+00
8       BsmtUnfSF  2.577553e+00
9     TotalBsmtSF  5.130743e+00
10     FirstFlrSF           inf
11    SecondFlrSF           inf
12   LowQualFinSF           inf
13      GrLivArea           inf
14   BsmtFullBath  2.242659e+00
15   BsmtHalfBath  1.140692e+00
16       FullBath  2.911473e+00
17       HalfBath  2.190673e+00
18   BedroomAbvGr  2.199007e+00
19   KitchenAbvGr  1.624357e+00
20   TotRmsAbvGrd  4.917930e+00
21     Fireplaces  1.556896e+00
22     GarageCars  5.565460e+00
23     GarageArea  5.397997e+00
24     WoodDeckSF  1.253248e+00
25    OpenPorchSF  1.219440e+00
26  EnclosedPorch  1.242530e+00
27       SsnPorch  1.032740e+00
28    ScreenPorch  1.127507e+00
```

```
29       PoolArea  1.095423e+00
30        MiscVal  1.032655e+00
31         MoSold  1.058330e+00
32         YrSold  1.048841e+00
```

```
C:\Users\akl0407\Anaconda3\lib\site-packages\statsmodels\stats\outliers_influence.py:193: Ru
  vif = 1. / (1. - r_squared_i)
```

```
#Remove another collinear predictor
remove = ['BsmtFinSF1','FirstFlrSF']
pred_filter = [x for x in predictors if x not in remove]
X = train[pred_filter]

#Recompute VIF
vif(X)
```

```
          feature           VIF
0           const  2.419858e+06
1      MSSubClass  1.480034e+00
2         LotArea  1.342832e+00
3     OverallQual  3.438243e+00
4     OverallCond  1.601301e+00
5       YearBuilt  3.965491e+00
6    YearRemodAdd  2.223979e+00
7      BsmtFinSF2  1.131439e+00
8       BsmtUnfSF  2.577553e+00
9     TotalBsmtSF  5.130743e+00
10    SecondFlrSF  6.338090e+00
11   LowQualFinSF  1.145700e+00
12      GrLivArea  1.085217e+01
13   BsmtFullBath  2.242659e+00
14   BsmtHalfBath  1.140692e+00
15       FullBath  2.911473e+00
16       HalfBath  2.190673e+00
17   BedroomAbvGr  2.199007e+00
18   KitchenAbvGr  1.624357e+00
19   TotRmsAbvGrd  4.917930e+00
20     Fireplaces  1.556896e+00
21     GarageCars  5.565460e+00
22     GarageArea  5.397997e+00
23     WoodDeckSF  1.253248e+00
```

```
24    OpenPorchSF  1.219440e+00
25  EnclosedPorch  1.242530e+00
26      SsnPorch  1.032740e+00
27    ScreenPorch  1.127507e+00
28       PoolArea  1.095423e+00
29        MiscVal  1.032655e+00
30         MoSold  1.058330e+00
31         YrSold  1.048841e+00
```

There is no more multicollinearity.

G.28 Lasso

Develop a lasso regression model to predict sale price based on all the predictors (except Id) in housing_train.csv. Find the RMSE (root mean squared error) of the developed model on housing_test.csv. Round up your answer to the nearest 100 greater than the answer. For example is the RMSE is 1001, enter 1100 in the box.

Note: Use this range of tuning parameter to find its optimal value:

```
alphas = 10**np.linspace(0,-4,200)*0.5
```

```
X = train[predictors]
#Test dataset
Xtest = test[predictors]
```

```
#Standardizing test data
y = np.log(train.SalePrice)
scaler = StandardScaler()
scaler.fit(X)
Xstd = scaler.transform(X)
Xtest_std = scaler.transform(Xtest)
```

```
alphas = 10**np.linspace(0,-4,200)*0.5

lassocv = LassoCV(alphas = alphas, cv = 10, max_iter = 100000)
lassocv.fit(Xstd, y)

#Optimal value of the tuning parameter - lamda
lassocv.alpha_
```

```
0.005359456596025638
```

```
#Using the developed lasso model to predict on test data
lasso = Lasso(alpha = lassocv.alpha_)
lasso.fit(Xstd, y)
```

```
Lasso(alpha=0.005359456596025638)
```

```
pred=np.exp(lasso.predict(Xtest_std))
np.sqrt(((pred-test.SalePrice)**2).mean())
```

```
25296.540649657465
```

G.29 Predictor importance

Which predictor is the most important in predicting sale price based on the lasso regression model (developed in Q28)?

Hint: Find the predictor with the highest magnitude of coefficient.

```
X.columns[np.argmax(np.abs(lasso.coef_))]
```

```
'OverallQual'
```

G.30 Improving model fit

Remove the influential points from the train data housing_train.csv. Re-develop the lasso regression model, and compute the RMSE (root mean squared error) of the developed model on housing_test.csv. Round up your answer to the nearest 100 greater than the answer. For example is the RMSE is 1001, enter 1100 in the box.

Note: Assume that a data point having a leverage more than 4 times the average leverage and a studentized residual with a magnitude of more than 3 is an influential point.

```
model_log = sm.ols('np.log(SalePrice)~' + '+'.join(predictors), data = train).fit()
out = model_log.outlier_test()
```

```
#Average leverage of points
average_leverage = (model_log.df_model+1)/model_log.nobs
average_leverage

#Computing the leverage statistic for each observation
influence = model_log.get_influence()
leverage = influence.hat_matrix_diag

#We will remove all observations that have leverage higher than the threshold value.
high_leverage_threshold = 4*average_leverage

#Number of high leverage points in the dataset
np.sum(leverage>high_leverage_threshold)
```

```
15
```

```
#Dropping influential points from data
train_filtered = train.drop(np.intersect1d(np.where(np.abs(out.student_resid)>3)[0],
                                           (np.where(leverage>high_leverage_threshold)[0])
```

```
X = train_filtered[predictors]
```

```
#Standardizing test data
y = np.log(train_filtered.SalePrice)
scaler = StandardScaler()
scaler.fit(X)
Xstd = scaler.transform(X)
Xtest_std = scaler.transform(Xtest)
```

```
alphas = 10**np.linspace(0,-4,200)*0.5

lassocv = LassoCV(alphas = alphas, cv = 10, max_iter = 100000)
lassocv.fit(Xstd, y)

#Optimal value of the tuning parameter - lamda
lassocv.alpha_
```

```
0.005117057010527266
```

```
#Using the developed lasso model to predict on test data
lasso = Lasso(alpha = lassocv.alpha_)
lasso.fit(Xstd, y)
```

```
Lasso(alpha=0.005117057010527266)
```

```
pred=np.exp(lasso.predict(Xtest_std))
np.sqrt(((pred-test.SalePrice)**2).mean())
```

```
22684.858174164117
```

H Datasets, assignment and project files

Datasets used in the book, assignment files, project files, and prediction problems report tempate can be found here

References