Tugas LAB WEEK 4 Christopher Darren

jupyter 00000054804_Christopher Darren_Week4 Last Checkpoint: 5 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

Data Mining Methodology 2

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

Linear Regression

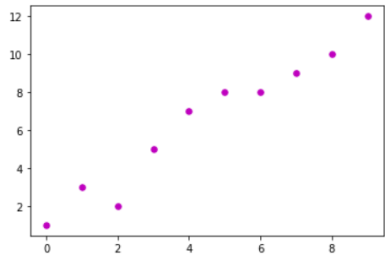
In [2]:
```
#independent_var merupakan variable yang mempengaruhi dependent_var
#independent_var sebagai data x, dependent_var sebagai data y

independent_var = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
dependent_var = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])
```

In [3]:
```
#number of observations/points
n = np.size(independent_var)
```

In [4]:
```
#menghitung rata - rata (mean) untuk menghitung cross deviation dan squared deviation

# mean of x and y vector
mean_indep_var = np.mean(independent_var)
mean_dep_var = np.mean(dependent_var)

#Fungsi cross deviation dan squared deviation adalah untuk membuat garis linear yang menggambarkan posisi
#dependent variable yang akan diprediksi

#calculating cross-deviation and deviation about x
cross_deviation = np.sum(dependent_var * independent_var) - n*mean_dep_var*mean_indep_var
squared_deviation = np.sum(independent_var*independent_var) - n*mean_indep_var*mean_indep_var
```

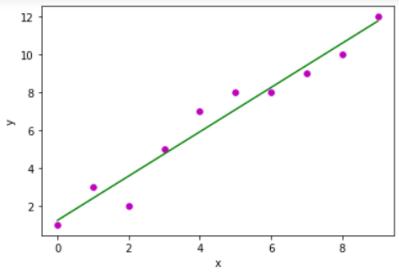
Gambar 1.

In [5]:
```
#calculating regression coefficients
b_1 = cross_deviation / squared_deviation
b_0 = mean_dep_var - b_1*mean_indep_var

#plotting the actual points as scatter plot
plt.scatter(independent_var, dependent_var, color = 'm', marker = "o", s= 30)

# predicted response vector
y_pred = b_0 + b_1*independent_var
```



In [6]:
```
# plotting the regression Line
plt.plot(independent_var, y_pred, color = 'g')

# putting labels
plt.xlabel('x')
plt.ylabel('y')

#function to show plot
plt.scatter(independent_var, dependent_var, color = "m", marker = "o", s = 30)
plt.show()
```

Gambar 2.



1. Penerapan Linear Regression pada Weather untuk memprediksi temperatur suhu

In [7]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

In [8]:
```
#reading data
dataset = pd.read_csv(r"D:\SEMESTER 4\IS411 Data Modelling\LAB\Bahan Modul 4\weather.csv", header=0)
dataset.info()
dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MinTemp        366 non-null    float64
 1   MaxTemp        366 non-null    float64
```

Gambar 3.

```
 10  WindSpeed3pm   366 non-null    int64
 11  Humidity9am    366 non-null    int64
 12  Humidity3pm    366 non-null    int64
 13  Pressure9am    366 non-null    float64
 14  Pressure3pm    366 non-null    float64
 15  Cloud9am       366 non-null    int64
 16  Cloud3pm       366 non-null    int64
 17  Temp9am        366 non-null    float64
 18  Temp3pm        366 non-null    float64
 19  RainToday      366 non-null    object
 20  RISK_MM        366 non-null    float64
 21  RainTomorrow   366 non-null    object
dtypes: float64(12), int64(5), object(5)
memory usage: 63.0+ KB
```

Out[8]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humidity3pm	Pressure
0	8.0	24.3	0.0	3.4	6.3	NW	30.0	SW	NW	6.0	...	29	10
1	14.0	26.9	3.6	4.4	9.7	ENE	39.0	E	W	4.0	...	36	10
2	13.7	23.4	3.6	5.8	3.3	NW	85.0	N	NNE	6.0	...	69	10
3	13.3	15.5	39.8	7.2	9.1	NW	54.0	WNW	W	30.0	...	56	10
4	7.6	16.1	2.8	5.6	10.6	SSE	50.0	SSE	ESE	20.0	...	49	10
...	...	...	...	...	...	...	...	...	...	...	...	...	
361	9.0	30.7	0.0	7.6	12.1	NNW	76.0	SSE	NW	7.0	...	15	10
362	7.1	28.4	0.0	11.6	12.7	N	48.0	NNW	NNW	2.0	...	22	10
363	12.5	19.9	0.0	8.4	5.3	ESE	43.0	ENE	ENE	11.0	...	47	10
364	12.5	26.9	0.0	5.0	7.1	NW	46.0	SSW	WNW	6.0	...	39	10
365	12.3	30.2	0.0	6.0	12.6	NW	78.0	NW	WNW	31.0	...	13	10

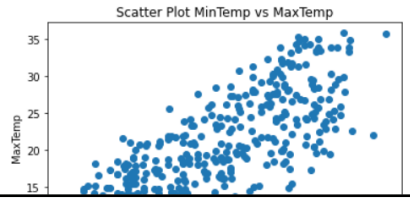
366 rows × 22 columns

Gambar 4.

In [9]:
```
#kita akan melakukan prediksi MaxTemp berdasarkan MinTemp
#maka MaxTemp sebagai dependent variable (y) dan MinTemp sebagai independent variable (x)
#kolom yang akan digunakan adalah MinTemp dan MaxTemp

dataTemp = dataset[['MinTemp','MaxTemp']]
dataTemp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   MinTemp  366 non-null    float64
 1   MaxTemp  366 non-null    float64
dtypes: float64(2)
memory usage: 5.8 KB
```

In [10]:
```
#melihat korelasi data dengan Scatter Plot

plt.scatter(dataTemp['MinTemp'], dataTemp['MaxTemp'])
plt.xlabel('MinTemp')
plt.ylabel('MaxTemp')
plt.title('Scatter Plot MinTemp vs MaxTemp')
plt.show()
```



Gambar 5

In [11]:
```
x = dataTemp['MinTemp'].values.reshape(-1, 1)
y = dataTemp['MaxTemp'].values.reshape(-1, 1)

#reshape (-1, 1) untuk mengubah susunan data menjadi row ke samping.
```

In [12]:
```
#membagi data training dan data testing

x_train, x_test , y_train, y_test = train_test_split(x, y, test_size=0.4)
```

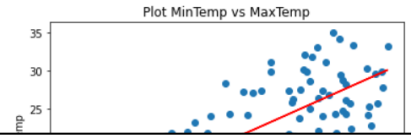
In [13]:
```
LinReg = LinearRegression()
LinReg.fit(x_train, y_train)
LinReg.score(x_test, y_test)
```

Out[13]:
```
0.5623752206448489
```

In [14]:
```
#hasil akurasi yang saya peroleh = 0.5623752206448489
```

In [15]:
```
y_predict = LinReg.predict(x_test)
plt.scatter(x_test, y_test)
plt.plot(x_test, y_predict, c='r')
plt.xlabel('MinTemp')
plt.ylabel('MaxTemp')
plt.title('Plot MinTemp vs MaxTemp')
```

Out[15]:
```
Text(0.5, 1.0, 'Plot MinTemp vs MaxTemp')
```



Gambar 6.

Logistic Regression

In [16]:

```
#copy dataset
dataLogistic = dataset.copy()
dataLogistic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MinTemp        366 non-null    float64
 1   MaxTemp        366 non-null    float64
 2   Rainfall       366 non-null    float64
 3   Evaporation    366 non-null    float64
 4   Sunshine       363 non-null    float64
 5   WindGustDir    363 non-null    object
 6   WindGustSpeed  364 non-null    float64
 7   WindDir9am     335 non-null    object
 8   WindDir3pm     365 non-null    object
 9   WindSpeed9am   359 non-null    float64
 10  WindSpeed3pm   366 non-null    int64
 11  Humidity9am    366 non-null    int64
 12  Humidity3pm    366 non-null    int64
 13  Pressure9am    366 non-null    float64
 14  Pressure3pm    366 non-null    float64
 15  Cloud9am       366 non-null    int64
 16  Cloud3pm       366 non-null    int64
 17  Temp9am        366 non-null    float64
 18  Temp3pm        366 non-null    float64
 19  RainToday      366 non-null    object
 20  RISK_MM        366 non-null    float64
 21  RainTomorrow   366 non-null    object
dtypes: float64(12), int64(5), object(5)
memory usage: 63.0+ KB
```

Gambar 7.

In [17]:

```
#mengatasi null
dataLogistic['WindGustSpeed'] = dataLogistic['WindGustSpeed'].fillna(0)
dataLogistic['WindSpeed9am'] = dataLogistic['WindSpeed9am'].fillna(0)
dataLogistic['Sunshine'] = dataLogistic['Sunshine'].fillna(0)
dataLogistic.info()
dataLogistic
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MinTemp        366 non-null    float64
 1   MaxTemp        366 non-null    float64
 2   Rainfall       366 non-null    float64
 3   Evaporation    366 non-null    float64
 4   Sunshine       366 non-null    float64
 5   WindGustDir    363 non-null    object
 6   WindGustSpeed  366 non-null    float64
 7   WindDir9am     335 non-null    object
 8   WindDir3pm     365 non-null    object
 9   WindSpeed9am   366 non-null    float64
 10  WindSpeed3pm   366 non-null    int64
 11  Humidity9am    366 non-null    int64
 12  Humidity3pm    366 non-null    int64
 13  Pressure9am    366 non-null    float64
 14  Pressure3pm    366 non-null    float64
 15  Cloud9am       366 non-null    int64
 16  Cloud3pm       366 non-null    int64
 17  Temp9am        366 non-null    float64
 18  Temp3pm        366 non-null    float64
 19  RainToday      366 non-null    object
 20  RISK_MM        366 non-null    float64
 21  RainTomorrow   366 non-null    object
dtypes: float64(12), int64(5), object(5)
memory usage: 63.0+ KB
```

Gambar 8

Out[17]:

	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	...	Humidity3pm	Pressure
0	8.0	24.3	0.0	3.4	6.3	NW	30.0	SW	NW	6.0	...	29	10
1	14.0	26.9	3.6	4.4	9.7	ENE	39.0	E	W	4.0	...	36	10
2	13.7	23.4	3.6	5.8	3.3	NW	85.0	N	NNE	6.0	...	69	10
3	13.3	15.5	39.8	7.2	9.1	NW	54.0	WNW	W	30.0	...	56	10
4	7.6	16.1	2.8	5.6	10.6	SSE	50.0	SSE	ESE	20.0	...	49	10
...	...	...	...	...	...	...	...	...	...	...	...	...	
361	9.0	30.7	0.0	7.6	12.1	NNW	76.0	SSE	NW	7.0	...	15	10
362	7.1	28.4	0.0	11.6	12.7	N	48.0	NNW	NNW	2.0	...	22	10
363	12.5	19.9	0.0	8.4	5.3	ESE	43.0	ENE	ENE	11.0	...	47	10
364	12.5	26.9	0.0	5.0	7.1	NW	46.0	SSW	WNW	6.0	...	39	10
365	12.3	30.2	0.0	6.0	12.6	NW	78.0	NW	WNW	31.0	...	13	10

366 rows × 22 columns

In [18]:

```
#encoding

def encode_data(feature_name):
    '''
    This function takes feature name as a parameter and returns mapping dictionary to replace(or map) categorical data with
    '''
    mapping_dict = {}

    unique_values = list(dataLogistic[feature_name].unique())


    for idx in range(len(unique_values)):
```

Gambar 9.

```
    return mapping_dict


dataLogistic['RainToday'].replace({'No':0,'Yes': 1}, inplace = True)

dataLogistic['RainTomorrow'].replace({'No':0,'Yes': 1}, inplace = True)

dataLogistic['WindGustDir'].replace(encode_data('WindGustDir'), inplace = True)

dataLogistic['WindDir9am'].replace(encode_data('WindDir9am'), inplace = True)

dataLogistic['WindDir3pm'].replace(encode_data('WindDir3pm'), inplace = True)
```

In [19]:

```
#checking data
dataLogistic
dataLogistic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 366 entries, 0 to 365
Data columns (total 22 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   MinTemp        366 non-null    float64
 1   MaxTemp        366 non-null    float64
 2   Rainfall       366 non-null    float64
 3   Evaporation    366 non-null    float64
 4   Sunshine       366 non-null    float64
 5   WindGustDir    366 non-null    float64
 6   WindGustSpeed  366 non-null    float64
 7   WindDir9am     366 non-null    float64
 8   WindDir3pm     366 non-null    float64
 9   WindSpeed9am   366 non-null    float64
 10  WindSpeed3pm   366 non-null    int64
 11  Humidity9am    366 non-null    int64
 12  Humidity3pm    366 non-null    int64
```

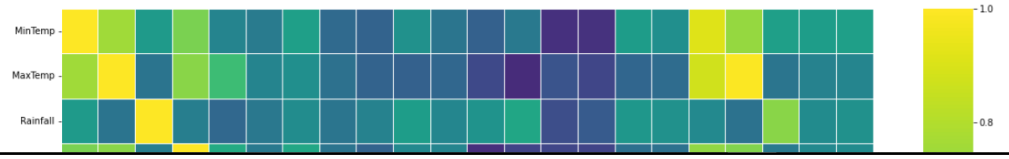
Gambar 10.

```
 13  Pressure9am    366 non-null    float64
 14  Pressure3pm    366 non-null    float64
 15  Cloud9am       366 non-null    int64
 16  Cloud3pm       366 non-null    int64
 17  Temp9am        366 non-null    float64
 18  Temp3pm        366 non-null    float64
 19  RainToday      366 non-null    int64
 20  RISK_MM        366 non-null    float64
 21  RainTomorrow   366 non-null    int64
dtypes: float64(15), int64(7)
memory usage: 63.0 KB
```

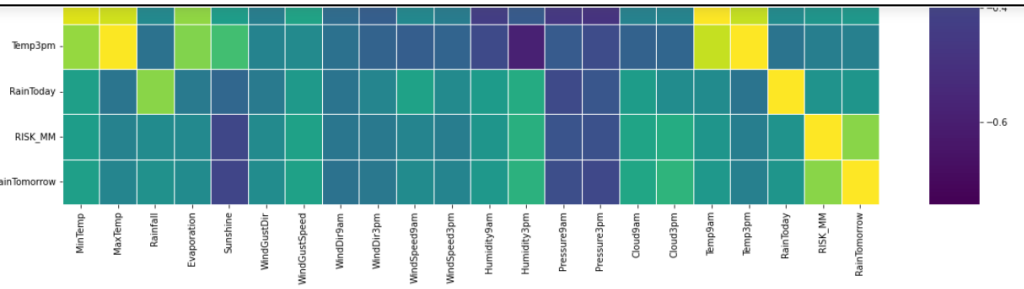
In [20]:
```
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
```

In [21]:
```
#visualisasi menggunakan heatmap untuk menemukan hubungan

plt.figure(figsize=(20,20))
sns.heatmap(dataLogistic.corr(), linewidth=0.5, annot=False, fmt=".2f", cmap = 'viridis')

#Linewidths untuk tebal garis; Linecolor untuk garis diluar visualisasi; annot untuk label tiap baris;
#cmap untuk pilih tema warna
#lihat pilihan warna di seaborn: https://seaborn.pydata.org/tutorial/color_palettes.html
```

Out[21]: <AxesSubplot:>



Gambar 11.



In [22]:
```
#menentukan nilai x dan y untuk prediksi turun hujan

X = dataLogistic.drop(['RainTomorrow'], axis= 1)
y = dataLogistic['RainTomorrow']

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.4, random_state = 0)

print("Length of Training Data: {}".format(len(X_train)))
print("Length of Testing Data: {}".format(len(X_test)))
```

```
Length of Training Data: 219
Length of Testing Data: 147
```

In [23]:
```
#memasukkan data ke dalam model Logistic Regression
logreg = LogisticRegression(solver='liblinear')
logreg.fit(X_train, y_train)
```

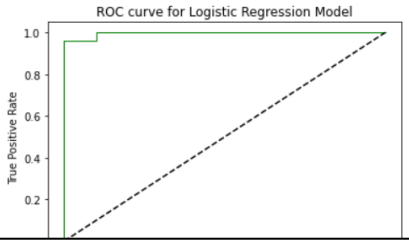
Out[23]: LogisticRegression(solver='liblinear')

Gambar 12.

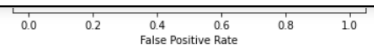
In [24]:

```
#evaluating data using ROC Curve
#hasil yang kemungkinan terjadi adalah

#True Positive: prediksi hujan, dan turun hujan
#True Negative: prediksi tidak ada huja, dan tidak turun hujan

print("Accuracy : ", logreg.score(X_test, y_test))

from sklearn. metrics import roc_curve

y_pred_logreg_proba = logreg.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_logreg_proba)
plt.figure(figsize=(6, 4))
plt.plot(fpr, tpr, '-g', linewidth=1)
plt.plot([0,1],[0,1], 'k--')
plt.title('ROC curve for Logistic Regression Model')
plt.xlabel("False Positive Rate")
plt.ylabel('True Positive Rate')
plt.show()
```

```
Accuracy :  0.9931972789115646
```



Gambar 13.



==Exercise Linear Regression dan Logistic Regression Terhadap harga_mobil==

2. Penerapan Linear Regression pada harga_mobil dalam memprediksi harga

In [25]:

```
#reading data
dataset2 = pd.read_csv(r"D:\SEMESTER 4\IS411 Data Modelling\LAB\Bahan Modul 4\harga_mobil.csv", header=0)
dataset2.info()
dataset2.head(5)
```

```
 23  city-mpg           201 non-null    int64
 24  highway-mpg        201 non-null    int64
 25  price              201 non-null    int64
dtypes: float64(10), int64(6), object(10)
memory usage: 41.0+ KB
```

Out[25]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepow
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111
1	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111
2	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154
3	2	164.0	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102

Gambar 14.

In [26]:
```
dataset2.describe()
```

Out[26]:

	symboling	normalized-losses	wheel-base	length	width	height	curb-weight	engine-size	bore	stroke	compression-ratio	horsepower
count	201.000000	164.000000	201.000000	201.000000	201.000000	201.000000	201.000000	201.000000	197.000000	197.000000	201.000000	199.000000
mean	0.840796	122.000000	98.797015	174.200995	65.889055	53.766667	2555.666667	126.875622	3.330711	3.256904	10.164279	103.396985
std	1.254802	35.442168	6.066366	12.322175	2.101471	2.447822	517.296727	41.546834	0.270793	0.319256	4.004965	37.553843
min	-2.000000	65.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	2.540000	2.070000	7.000000	48.000000
25%	0.000000	94.000000	94.500000	166.800000	64.100000	52.000000	2169.000000	98.000000	3.150000	3.110000	8.600000	70.000000
50%	1.000000	115.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	3.310000	3.290000	9.000000	95.000000
75%	2.000000	150.000000	102.400000	183.500000	66.600000	55.500000	2926.000000	141.000000	3.590000	3.410000	9.400000	116.000000
max	3.000000	256.000000	120.900000	208.100000	72.000000	59.800000	4066.000000	326.000000	3.940000	4.170000	23.000000	262.000000

In [27]:
```
dataset2.isnull().sum()
```

Out[27]:
```
symboling             0
normalized-losses    37
make                  0
fuel-type             0
aspiration            0
num-of-doors          2
body-style            0
drive-wheels          0
engine-location       0
wheel-base            0
length                0
width                 0
height                0
curb-weight           0
```

Gambar 15.

```
num-of-doors          2
body-style            0
drive-wheels          0
engine-location       0
wheel-base            0
length                0
width                 0
height                0
curb-weight           0
engine-type           0
num-of-cylinders      0
engine-size           0
fuel-system           0
bore                  4
stroke                4
compression-ratio     0
horsepower            2
peak-rpm              2
city-mpg              0
highway-mpg           0
price                 0
dtype: int64
```

In [28]:
```
def clean_dataset(dataset2):
    assert isinstance(dataset2, pd.DataFrame), "dataset2 needs to be a pd.DataFrame"
    dataset2.dropna(inplace=True)
    indices_to_keep = ~dataset2.isin([np.nan, np.inf, -np.inf]).any(axis=1)
    return dataset2[indices_to_keep].astype(np.float64)
```

In [29]:
```
#get rid of infinite values.
dataset2.replace([np.inf, -np.inf], np.nan, inplace=True)
```

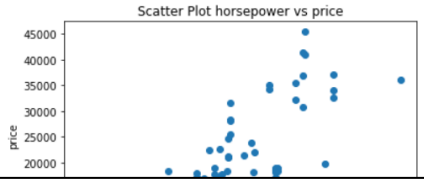
In [30]:
```
#fill missing values
dataset2.fillna(0, inplace=True)
```

Gambar 16.

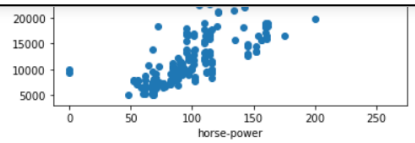
In [31]:
```
#kita akan melakukan prediksi price berdasarkan engine-size(cc)
#maka price sebagai dependent variable (y) dan horsepower sebagai independent variable (x)
#kolom yang akan digunakan adalah horsepower dan price

dataMobil = dataset2[['horsepower','price']]
dataMobil.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   horsepower  201 non-null    float64
 1   price       201 non-null    int64
dtypes: float64(1), int64(1)
memory usage: 3.3 KB
```

In [32]:
```
#melihat korelasi data dengan Scatter Plot

plt.scatter(dataMobil['horsepower'], dataMobil['price'])
plt.xlabel('horse-power')
plt.ylabel('price')
plt.title('Scatter Plot horsepower vs price')
plt.show()
```



Gambar 17.



In [33]:
```
x = dataMobil['horsepower'].values.reshape(-1, 1)
y = dataMobil['price'].values.reshape(-1, 1)

#reshape (-1, 1) untuk mengubah susunan data menjadi row ke samping.
```

In [34]:
```
#membagi data training dan data testing untuk harga_mobil

x_train, x_test , y_train, y_test = train_test_split(x, y, test_size=0.2)
```

In [35]:
```
LinReg2 = LinearRegression()
LinReg2.fit(x_train, y_train)
LinReg2.score(x_test, y_test)
```

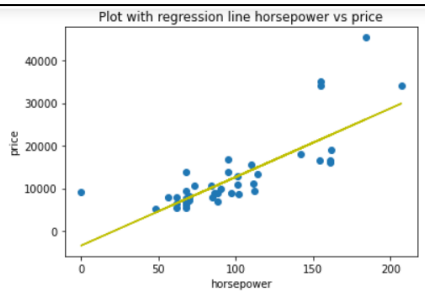
Out[35]:
```
0.6253679708642874
```

In [36]:
```
#hasil akurasi  train data mobil yang saya peroleh = 0.6253679708642874
```

In [37]:
```
y_predict = LinReg2.predict(x_test)
plt.scatter(x_test, y_test)
plt.plot(x_test, y_predict, c='y')
plt.xlabel('horsepower')
plt.ylabel('price')
plt.title('Plot with regression line horsepower vs price')
```

Out[37]:
```
Text(0.5, 1.0, 'Plot with regression line horsepower vs price')
```

Gambar 18.



Logistic Regression

In [38]:
```
#copy dataset mobil
dataLogistic2 = dataset2.copy()
dataLogistic2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          201 non-null    int64
 1   normalized-losses  201 non-null    float64
 2   make               201 non-null    object
 3   fuel-type          201 non-null    object
 4   aspiration         201 non-null    object
 5   num-of-doors       201 non-null    object
```

Gambar 19.

```
 6   body-style         201 non-null    object
 7   drive-wheels       201 non-null    object
 8   engine-location    201 non-null    object
 9   wheel-base         201 non-null    float64
 10  length             201 non-null    float64
 11  width              201 non-null    float64
 12  height             201 non-null    float64
 13  curb-weight        201 non-null    int64
 14  engine-type        201 non-null    object
 15  num-of-cylinders   201 non-null    object
 16  engine-size        201 non-null    int64
 17  fuel-system        201 non-null    object
 18  bore               201 non-null    float64
 19  stroke             201 non-null    float64
 20  compression-ratio  201 non-null    float64
 21  horsepower         201 non-null    float64
 22  peak-rpm           201 non-null    float64
 23  city-mpg           201 non-null    int64
 24  highway-mpg        201 non-null    int64
 25  price              201 non-null    int64
dtypes: float64(10), int64(6), object(10)
memory usage: 41.0+ KB
```

In [39]:
```
#encoding

def encode_data(feature_name):
    '''
    This function takes feature name as a parameter and returns mapping dictionary to replace(or map) categorical data with
    '''
    mapping_dict = {}

    unique_values = list(dataLogistic2[feature_name].unique())


    for idx1 in range(len(unique_values)):
```

Gambar 20.

```
        mapping_dict[unique_values[idx1]] = idx1

    return mapping_dict


#dataLogistic2['RainToday'].replace({'No':0,'Yes': 1}, inplace = True)

#dataLogistic2['RainTomorrow'].replace({'No':0,'Yes': 1}, inplace = True)

dataLogistic2['make'].replace(encode_data('make'), inplace = True)

dataLogistic2['fuel-type'].replace(encode_data('fuel-type'), inplace = True)

dataLogistic2['aspiration'].replace(encode_data('aspiration'), inplace = True)

dataLogistic2['num-of-doors'].replace(encode_data('num-of-doors'), inplace = True)

dataLogistic2['body-style'].replace(encode_data('body-style'), inplace = True)

dataLogistic2['drive-wheels'].replace(encode_data('drive-wheels'), inplace = True)

dataLogistic2['engine-location'].replace(encode_data('engine-location'), inplace = True)

dataLogistic2['engine-type'].replace(encode_data('engine-type'), inplace = True)

dataLogistic2['num-of-cylinders'].replace(encode_data('num-of-cylinders'), inplace = True)

dataLogistic2['fuel-system'].replace(encode_data('fuel-system'), inplace = True)
```

In [40]:

```
#checking the changes
dataLogistic2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 201 entries, 0 to 200
Data columns (total 26 columns):
```

Gambar 21.

```
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   symboling          201 non-null    int64
 1   normalized-losses  201 non-null    float64
 2   make               201 non-null    int64
 3   fuel-type          201 non-null    int64
 4   aspiration         201 non-null    int64
 5   num-of-doors       201 non-null    int64
 6   body-style         201 non-null    int64
 7   drive-wheels       201 non-null    int64
 8   engine-location    201 non-null    int64
 9   wheel-base         201 non-null    float64
 10  length             201 non-null    float64
 11  width              201 non-null    float64
 12  height             201 non-null    float64
 13  curb-weight        201 non-null    int64
 14  engine-type        201 non-null    int64
 15  num-of-cylinders   201 non-null    int64
 16  engine-size        201 non-null    int64
 17  fuel-system        201 non-null    int64
 18  bore               201 non-null    float64
 19  stroke             201 non-null    float64
 20  compression-ratio  201 non-null    float64
 21  horsepower         201 non-null    float64
 22  peak-rpm           201 non-null    float64
 23  city-mpg           201 non-null    int64
 24  highway-mpg        201 non-null    int64
 25  price              201 non-null    int64
dtypes: float64(10), int64(16)
memory usage: 41.0 KB
```

In [41]:

```
def clean_dataset(dataLogistic2):
    assert isinstance(dataLogistic2, pd.DataFrame), "dataset2 needs to be a pd.DataFrame"
    dataLogistic2.dropna(inplace=True)
    indices_to_keep = ~dataLogistic2.isin([np.nan, np.inf, -np.inf]).any(axis=1)
    return dataLogistic2[indices_to_keep].astype(np.float64)
```

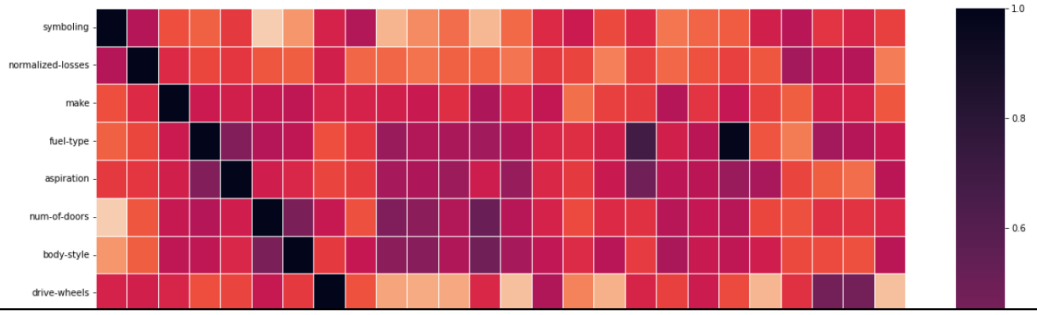
Gambar 22.

In [42]:
```
#get rid of infinite values.
dataLogistic2.replace([np.inf, -np.inf], np.nan, inplace=True)
```

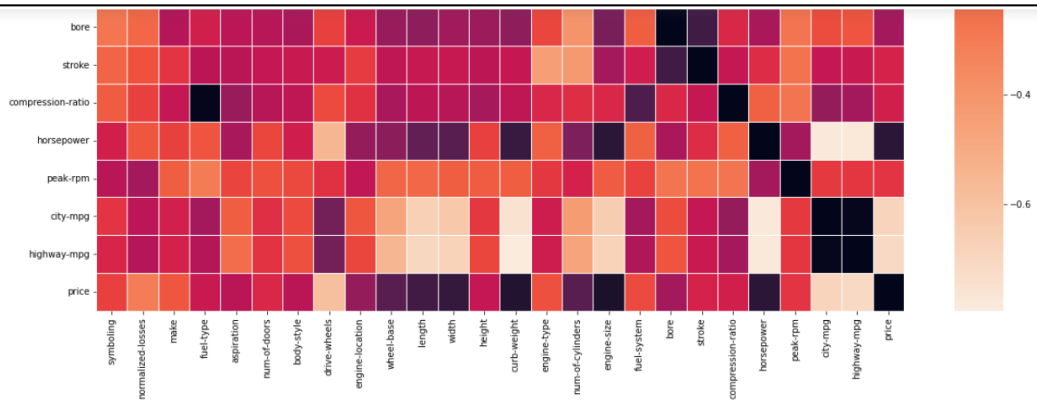
In [43]:
```
#fill missing values
dataLogistic2.fillna(0, inplace=True)
```

In [44]:
```
#visualisasi menggunakan heatmap untuk menemukan hubungan

plt.figure(figsize=(20,20))
sns.heatmap(dataLogistic2.corr(), linewidth=0.5, annot=False, fmt=".2f", cmap = 'rocket_r')

#Linewidths untuk tebal garis; Linecolor untuk garis diluar visualisasi; annot untuk label tiap baris;
#cmap untuk pilih tema warna
#lihat pilihan warna di seaborn: https://seaborn.pydata.org/tutorial/color_palettes.html
```

Out[44]: <AxesSubplot:>



Gambar 23.



In [45]:
```
#menentukan nilai x dan y untuk prediksi harga mobil

X = dataLogistic2.drop(['price'], axis= 1)
y = dataLogistic2['price']

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2, random_state = 0)

print("Length of Training Data: {}".format(len(X_train)))
print("Length of Testing Data: {}".format(len(X_test)))
```

```
Length of Training Data: 160
Length of Testing Data: 41
```

Gambar 24.

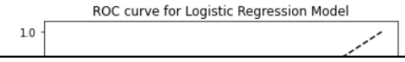
In [46]:
```
#memasukkan data ke dalam model Logistic Regression
logreg2 = LogisticRegression(solver='liblinear')
logreg2.fit(X_train, y_train)
```

Out[46]:
```
LogisticRegression(solver='liblinear')
```

In [47]:
```
#evaluating data using ROC Curve
#hasil yang kemungkinan terjadi adalah

#True Positive: prediksi harga naik, dan harga turun
#True Negative: prediksi harga tidak naik, dan harga tidak turun

print("Accuracy : ", logreg2.score(X_test, y_test))

from sklearn. metrics import roc_curve

y_pred_logreg2_proba = logreg2.predict_proba(X_test)[:,1]
fpr1, tpr1, thresholds = roc_curve(y_test, y_pred_logreg2_proba , pos_label=1)
plt.figure(figsize=(6, 4))
plt.plot(fpr1, tpr1, '-g', linewidth=1)
plt.plot([0,1],[0,1], 'k--')
plt.title('ROC curve for Logistic Regression Model')
plt.xlabel("False Positive Rate")
plt.ylabel('True Positive Rate')
plt.show()
```

```
Accuracy :  0.024390243902439025
```

```
C:\Users\Darren\anaconda3\lib\site-packages\sklearn\metrics\_ranking.py:999: UndefinedMetricWarning: No positive samples in y_t
rue, true positive value should be meaningless
  warnings.warn(
```



Gambar 25.

C:\Users\Darren\anaconda3\lib\site-packages\sklearn\metrics_ranking.py:999: UndefinedMetricWarning: No positive samples in y_t
rue, true positive value should be meaningless
  warnings.warn(

In [ ]:
```
1
```

Gambar 26.