Practical - 14

AIM

Implement a classification/ logistic regression problem.

PROBLEM

Based on age of a person predict whether a person will buy insurance or not.

CODE & OUTPUT

In [ ]:
```
# import statements
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

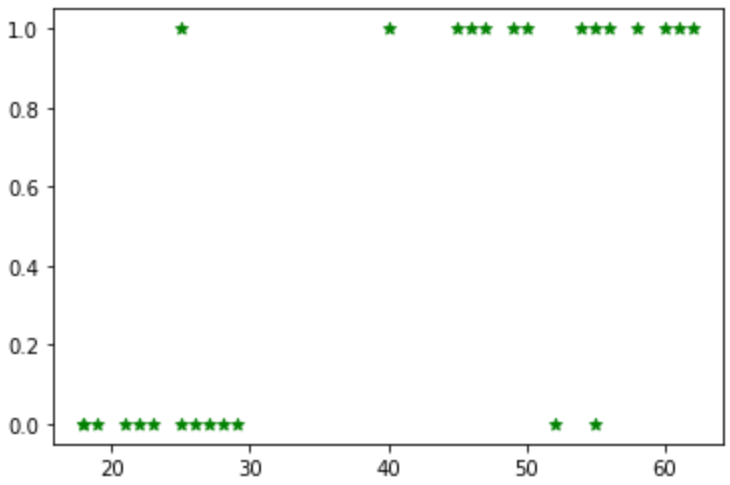
In [4]:
```
# read the data file
df = pd.read_csv("insurance_data.csv")
df.head()
```

Out[4]:

	age	bought_insurance
0	22	0
1	25	0
2	47	1
3	52	0
4	46	1

In [5]:
```
# scatter plot the data
plt.scatter(df.age, df.bought_insurance, marker="*", color="green")
```

Out[5]:
```
<matplotlib.collections.PathCollection at 0x1700f427160>
```



In [7]:
```
# split the data into 2 parts: train data & test data
from sklearn.model_selection import train_test_split
```

```
x_train, x_test, y_train, y_test = train_test_split(df[['age']], df.bought_insurance, trai
```

In [8]:
```
x_test
```

Out[8]:

	age
2	47
4	46
1	25
21	26
25	54
5	56

In [9]:
```
# build a Logistic Regression model
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [10]:
```
# fit the training data on the model
model.fit(x_train, y_train)
```

Out[10]:
```
LogisticRegression()
```

In [11]:
```
# predict y for the test data
y_pred = model.predict(x_test)
print(y_pred)
```

```
[1 1 0 0 1 1]
```

In [13]:
```
# get the coeffient in the equation y = mx + c
model.coef_
```

Out[13]:
```
array([[0.11332417]])
```

In [14]:
```
# get the intercept in the equation y = mx + c
model.intercept_
```

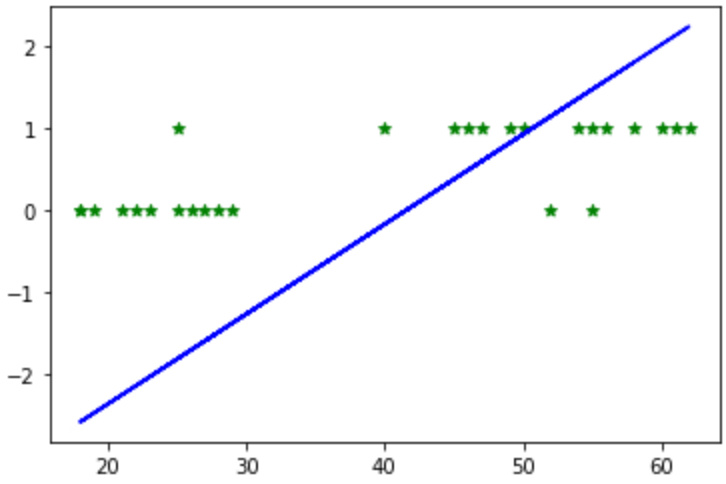
Out[14]:
```
array([-4.56853185])
```

In [15]:
```
# get the score of model on test set
model.score(x_test, y_test)
```

Out[15]:
```
1.0
```

In [16]:
```
# the best fit line can be written as
y = 0.11 * df.age - 4.57
plt.plot(df.age, y, color="blue")
plt.scatter(df.age, df.bought_insurance, marker="*", color="green")
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x17011f7cac0>
```



In [17]:
```
# predict whether a person of age 35 buy life insurance or not
res = model.predict([[35]])
```

In [18]:
```
res
```

Out[18]:
```
array([0], dtype=int64)
```

In [19]:
```
# predict whether a person of age 23 buy life insurance or not
res1 = model.predict([[23]])
res1
```

Out[19]:
```
array([0], dtype=int64)
```

In [20]:
```
# predict whether a person of age 65 buy life insurance or not
res2 = model.predict([[65]])
res2
```

Out[20]:
```
array([1], dtype=int64)
```