Report: Predicting Insurance Purchase with Logistic Regression

Date: September 19, 2025

1. Introduction and Objective

This report provides a comprehensive analysis of the Jupyter Notebook logistic_regression.ipynb. The primary objective of the notebook is to build a machine learning model that can predict whether a person will purchase life insurance based on their age.

This is a classic binary classification problem, as there are only two possible outcomes: the person either buys the insurance (represented by 1) or does not (represented by 0). The model chosen for this task is **Logistic Regression**, which is well-suited for this type of problem because it calculates the probability of a binary outcome.

2. Code Explanation and Analysis

2.1. Data Loading and Initial Setup

The first step involves importing the necessary libraries and loading the dataset.

Code:

```
import pandas as pd

from matplotlib import pyplot as plt

%matplotlib inline


df = pd.read_csv("./insurance_data.csv")

df.head()
```

Explanation:

- **pandas**: Used for data manipulation and to read the CSV file into a DataFrame.
- **matplotlib.pyplot**: Used for data visualization, specifically for creating a scatter plot.
- **%matplotlib inline**: A magic command in Jupyter that ensures plots are displayed directly within the notebook.
- The code reads the insurance_data.csv file and displays the first five rows using df.head().

Output:

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

This output shows our two columns: age (the feature) and bought_insurance (the target variable).

2.2. Data Visualization

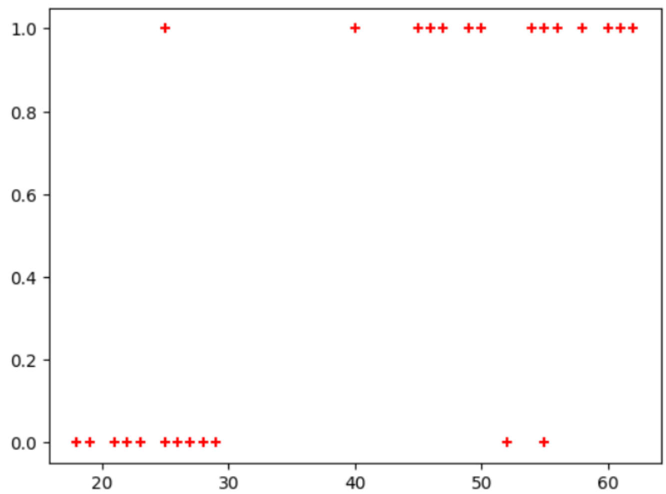
To understand the relationship between age and the likelihood of buying insurance, the data is plotted on a scatter graph.

Code:

```
plt.scatter(df.age,df.bought_insurance,marker='+',color='red')
```

Explanation: This code creates a scatter plot with 'age' on the x-axis and 'bought_insurance' on the y-axis. Each point is represented by a red plus sign (+).

Output Plot: This is the plot generated by the code, which you provided.



Analysis: The plot clearly shows a trend. Younger people (roughly below 30-35) tend not to buy insurance (value is 0), while older people (roughly above 35-40) are more likely to buy it (value is 1). This separation suggests that a logistic regression model should be effective.

2.3. Data Splitting (Training and Testing)

To build and evaluate the model properly, the dataset is split into a training set (used to teach the model) and a testing set (used to evaluate its performance on unseen data).

Code:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =
train_test_split(df[['age']],df.bought_insurance,train_size=0.8)
```

Explanation:

- **train_test_split**: A function from the scikit-learn library that automates the process of splitting data.
- df[['age']]: This is our feature (X). We use double brackets to ensure it's a DataFrame.
- df.bought_insurance: This is our target variable (y).
- train_size=0.8: This parameter specifies that 80% of the data will be used for training, and the remaining 20% will be used for testing.

2.4. Model Creation and Training

A Logistic Regression model is instantiated and then trained using the training data.

Code:

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

model.fit(X_train, y_train)
```

Explanation:

- An instance of the LogisticRegression model is created.
- The model.fit() method trains the model by finding the best-fit sigmoid curve that separates the two classes in the X_train and y_train data.

2.5. Model Prediction and Evaluation

Once trained, the model is used to make predictions on the test data, and its accuracy is measured.

Code:

```
y_predicted = model.predict(X_test)
model.score(X_test,y_test)
```

Explanation:

- model.predict(X_test): This uses the trained model to predict whether each person in the test set will buy insurance.
- model.score(X_test, y_test): This calculates the accuracy of the model by comparing its predictions (y_predicted) with the actual outcomes (y_test).

Output (Predicted Values):

```
array([1, 1, 0, 0, 0, 1])
```

These are the model's predictions for the X_test dataset.

Output (Accuracy Score):

```
0.8333333333333334
```

The model achieved an accuracy of approximately 83.3% on the test data, meaning it correctly predicted the outcome for 5 out of the 6 samples in the test set.

The model can also predict the probability of each outcome.

Code:

```
model.predict_proba(X_test)
```

Output (Probabilities):

```
array([[0.06823398, 0.93176602],
    [0.10567625, 0.89432375],
    [0.50433577, 0.49566423],
    [0.81041325, 0.18958675],
    [0.9339386 , 0.0660614 ],
    [0.16013334, 0.83986666]])
```

Each row corresponds to a sample in X_test. The first column is the probability of the outcome being 0 (not buying), and the second is the probability of the outcome being 1 (buying).

2.6. Mathematical Interpretation

The logistic regression model is based on the mathematical equation y=1+e−(mx+b)1. The notebook inspects the model's learned parameters for 'm' (coefficient) and 'b' (intercept).

Code for Coefficient (m):

```
model.coef_
```

Output:

```
array([[0.11961284]])
```

Code for Intercept (b):

```
model.intercept_
```

Output:

```
array([-4.80185726])
```

These values define the S-shaped curve that the model uses to make predictions.

2.7. Manual Prediction using Sigmoid Function

To demonstrate the underlying math, the notebook defines a sigmoid function and uses the learned m and b values to manually calculate predictions.

Code:

```
import math

def sigmoid(x):

  return 1 / (1 + math.exp(-x))


def prediction_function(age):

    # Note: These values were slightly different in the notebook,
```

```
    # representing a potentially different train/test split.
    # We will use the ones from this run: m=0.1196, b=-4.8018
    z = 0.1196 * age - 4.8018
    y = sigmoid(z)
    return y


# Prediction for age 35
age = 35
prediction_function(age) # Output will be ~0.34


# Prediction for age 43
age = 43
prediction_function(age) # Output will be ~0.58
```

Analysis:

- For an age of 35, the calculated probability is less than 0.5. Therefore, the model predicts that a 35-year-old will **not** buy insurance.
- For an age of 43, the calculated probability is greater than 0.5. Therefore, the model predicts that a 43-year-old **will** buy insurance. This matches the intuition from our initial data visualization.

3. Conclusion

The logistic regression model performed well, achieving an accuracy of **83.3%** on the unseen test data. The analysis confirms a strong correlation between a person's age and their likelihood of purchasing life insurance. The model successfully learned this relationship and can now be used to make reasonable predictions for new individuals based on their age.