Proceedings of IAM, V.14, N.2, 2025, pp.234-244
https://doi.org/10.30546/2225-0530.14.2.2025.2023

DATA VISUALIZATION USING MACHINE LEARNING APPLICATIONS

Ilaha Zafar Zafarli[1]

[1]Oilgasscientificresearchproject" Institute of State Oil Company of Azerbaijan Republic, Baku, Azerbaijan

axmedova_Ilaxa@mail.ru

Abstract: Artificial intelligence is a broad, multidisciplinary field of computer science concerned with the creation of hardware or software systems capable of performing tasks that normally require human intelligence. Artificial intelligence has a wide range of applications, from automating solutions to problems that require significant human effort in everyday life to the development of complex systems capable of making decisions based on large volumes of data. Currently, machine learning, one of the subfields of artificial intelligence, is widely used in solving various problems. This article examines the application of the following popular machine learning algorithms: linear regression, logistic regression, and the K-Nearest Neighbors (K-NN) algorithm. Therefore, this article first explains the essence of the linear regression method, one of the fundamental data analysis techniques, along with its implementation algorithm. Subsequently, the operating principle of the K-Nearest Neighbors (K-NN) algorithm is presented in detail.

Keywords: artificial intelligence, machine learning, data visualization.

AMS Subject Classification: 0308, 0311, 3910.

1. Introduction

Machine learning is one of the core concepts of artificial intelligence and plays an important role in solving data-driven problems in various application domains, including engineering, finance, and information systems [3,7]. In particular, supervised learning techniques are widely used for prediction and classification tasks due to their interpretability and ease of implementation.

Regression-based methods such as linear regression and logistic regression have been extensively studied and remain fundamental tools for numerical prediction and binary classification problems [1,3,5,9]. In addition to regression approaches, instance-based learning methods, including the K-Nearest Neighbors (K-NN) algorithm, have shown strong performance in classification tasks, especially when dealing with complex and non-linear data distributions [2,10,11].

Despite the widespread adoption of these algorithms, systematic comparison of their performance across different problem types and datasets is still a relevant research challenge [8]. Therefore, this paper aims to analyze and compare linear regression, logistic regression, and the K-Nearest Neighbors algorithm in terms of their performance and applicability.

2. Problem Solving Methods

In this study, problem solving is performed using supervised machine learning techniques, where historical data is used to train predictive models. The process includes data preprocessing, feature selection, model training, testing, and evaluation. Regression methods are applied for prediction tasks, while classification algorithms are used to determine categorical outcomes based on data similarity and decision boundaries.

Linear Regression – Linear regression is used to predict data characterized by numerical values. Some types of regression analysis are more suitable than others for processing more complex datasets [2,11,12]. Linear regression is an analytical method that predicts unknown values of one variable based on known values of another related variable. Mathematically, it models the relationship between a dependent (unknown) variable and an independent (known) variable as a linear equation. Linear regression is used to describe the relationship between data points by fitting a straight line through them. This line can then be used to predict unknown future values of the data. Simple linear regression is defined by the following linear function:

$$y = \beta_0 * x + \beta_1 + \varepsilon,$$

,

where β_0 and $\beta_1$are unknown constants representing the regression coefficients, and ε denotes the error term.

The linear regression of a dependent variable y on a set of independent variables $x = (x_1, \dots, x_r)$, where r is the number of predictors, defines a linear relationship between y and x:

$$y = \beta_0 + \beta_1 x_1 + \dots + \beta_r x_r + \varepsilon..$$

This is known as the regression equation, where β_0, β_1, …, β_r are the regression coefficients and ε is the random error.

Linear regression computes estimate of the regression coefficients b_0, b_1, …, b_r, also referred to as predicted measurement weights. These coefficients define the regression estimation function:

$$f(x) = b_0 + b_1 x_1 + \dots + b_r x_r \ ,$$

which adequately captures the dependency between inputs and outputs.

For each observation i = 1, …, n,, the estimated or predicted response $f(x_i)$ should be as close as possible to the actual response y_i. The differences $y_i - f(x_i)$ are called residuals. Regression determines the optimal predicted weights that minimize these residuals.

To obtain the best weights, the sum of squared residuals (SSR) must be minimized:

$$SSR = \Sigma_i (y_i - f(x_i))^2.$$

This approach is known as the least squares method. The implementation of simple linear regression begins with given input–output (x–y) pairs, which represent observation results. The estimated regression function is expressed as f(x)

= $b_0 + b_1x$. To minimize SSR and determine the regression estimation function, the optimal values of the predicted weights b_0 and b_1 must be calculated. The intercept b_0 indicates the point where the estimated regression line intersects the y-axis, while b_1 determines the slope of the regression line.

Residuals are calculated as $y_i - f(x_i) = y_i - b_0 - b_1x_i$ and represent the vertical distances between observed and predicted points. In linear regression, these distances must be minimized.

Simple or univariate linear regression, which involves a single independent variable, is illustrated in Figure 1. Essentially, the method attempts to construct a linear graph between two variables, x and y (Figure 2).

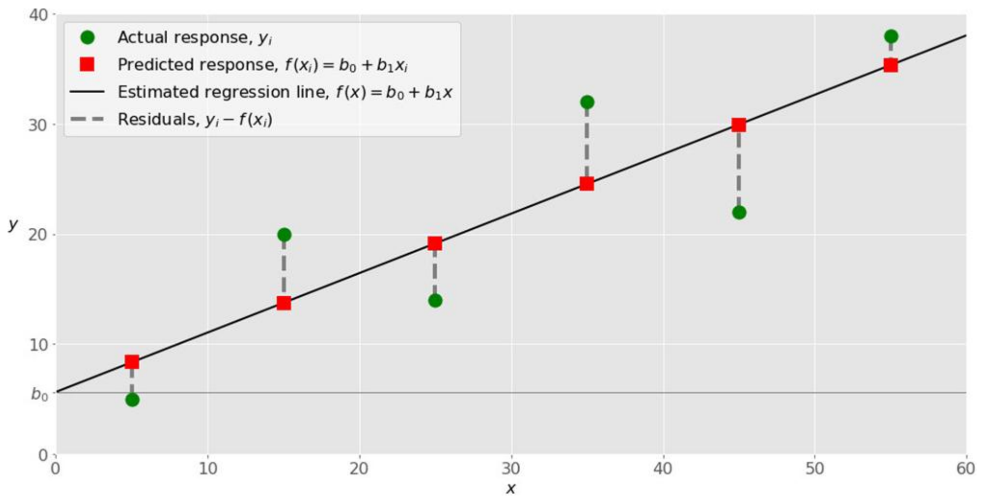


Figure 1. Univariate linear regression

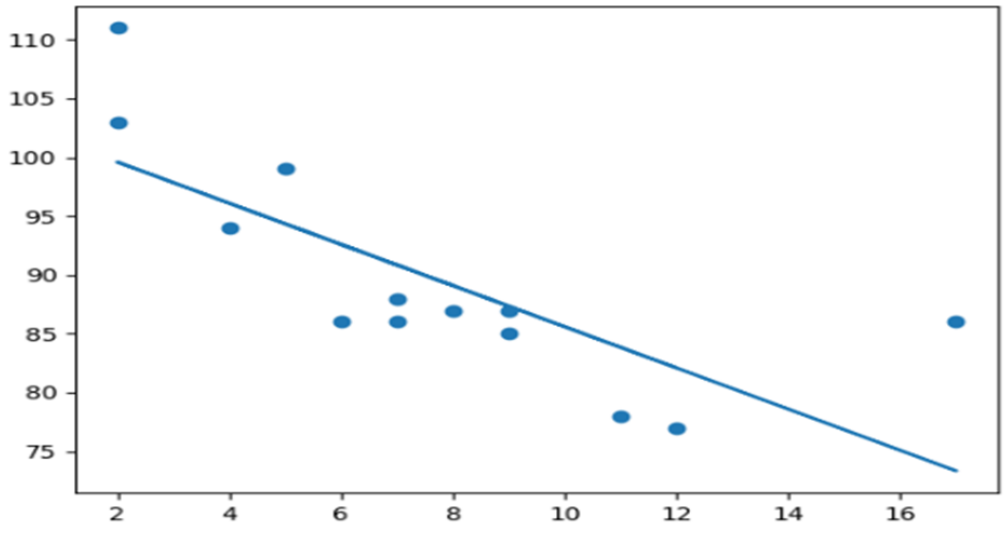


Figure 2. A line graph between two given variables

Understanding the relationship between the values on the X and Y axes is crucial [2,3,11,12]. If no relationship exists, linear regression cannot be used for prediction. This relationship is quantified by the correlation coefficient.

The implementation of linear regression generally involves the following five steps:

1. Importing the required packages and classes.
2. Providing and transforming the data.
3. Creating the regression model and fitting it to the available data.
4. Evaluating the adequacy and quality of model.
5. Applying the model to make predictions.

These steps are common to most regression approaches.

To determine the relationship between data points and draw the regression line in Python, the following libraries were used:

NumPy – a fundamental open-source scientific package for fast operations on one-dimensional and multi-dimensional arrays.

Scikit-learn – a widely used open-source machine learning library that provides tools for data preprocessing, dimensionality reduction, regression, classification, clustering, and more.

Logistic regression – Logistic regression is used for binary classification problems where the output variable has two logical values [1,6,7,10].

K-Nearest Neighbors (K-NN)- The K-Nearest Neighbors (K-NN) algorithm is used for both classification and regression tasks [4,5]. It is one of the simplest supervised machine learning algorithms and is based on the assumption that objects close to each other in feature space tend to have similar target values or belong to the same class.

The K-NN algorithm determines similarity between new data and existing data and assigns the new data point to the category most similar to it. The algorithm stores all training data and classifies new data points based on similarity. This allows new data to be easily classified as it appears. Although K-NN can be used for both regression and classification, it is more commonly applied to classification problems. K-NN is a non-parametric algorithm, meaning it makes no assumptions about the underlying data distribution. It is also referred to as a lazy learner because it does not learn immediately from the training set but instead stores the dataset and performs computation during classification.

To illustrate the necessity of the K-NN algorithm, consider a problem with two categories, A and B, and a new data point x_1. The task is to determine which category x_1 belongs to. K-NN can effectively solve such problems by identifying the category of a given data point based on proximity Figure 3.

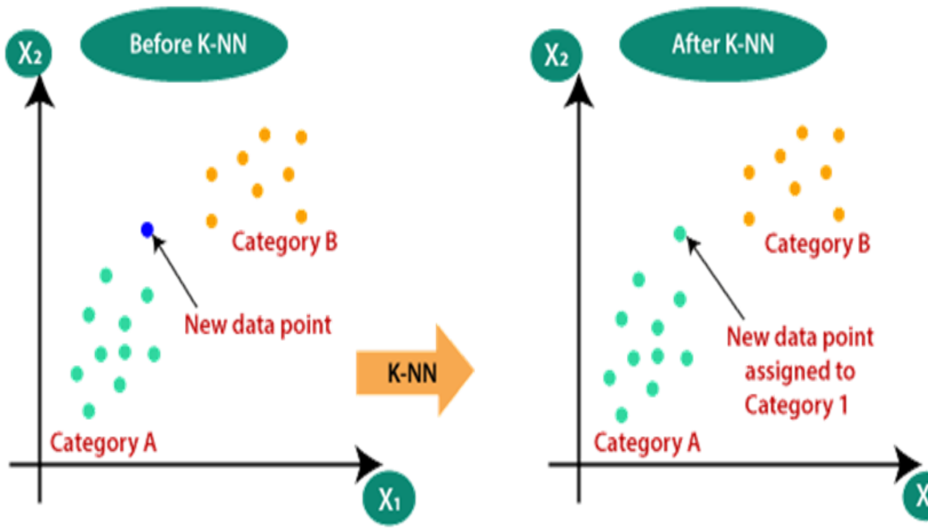


Figure 3. Determining the affiliation of data point x1 between categories A and B

The working principle of the K-NN algorithm consists of the following steps:

Step 1. Selecting the value of K.

Step 2. Calculating the Euclidean distances between the new data point and all existing data points.

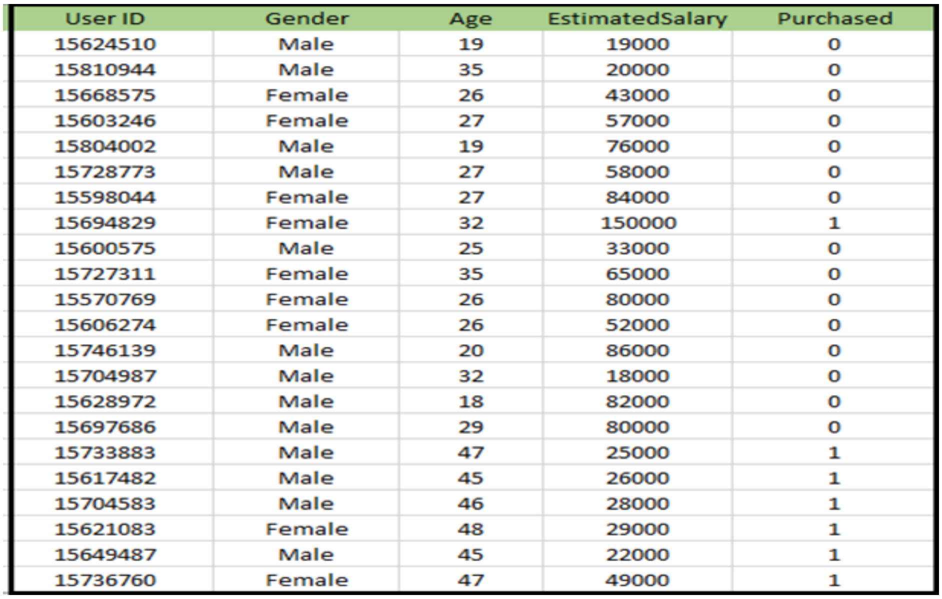
Step 3. Selecting the K nearest neighbors based on distance.

Step 4. Counting the number of data points belonging to each category among the K neighbors.

Step 5. Assigning the new data point to the category with the maximum number of neighbors.

Step 6. Completing the model construction.

Next, we examine the implementation of the K-Nearest Neighbors (K-NN) algorithm using the Python programming language. Considerable research has been conducted in this area [8,9]. In this study, the K-NN algorithm is implemented using the same dataset that was previously applied for logistic regression. The dataset consists of user information collected from social networks (Figure 4), where the independent variables are Age and Estimated Salary, and the dependent variable represents purchasing behavior. The specific origin of the dataset or the platform from which it was obtained is not of primary importance, as the dataset is used solely for the purpose of conducting subsequent analyses and was randomly selected from an Internet-based source [12].



User ID	Gender	Age	EstimatedSalary	Purchased
15624510	Male	19	19000	0
15810944	Male	35	20000	0
15668575	Female	26	43000	0
15603246	Female	27	57000	0
15804002	Male	19	76000	0
15728773	Male	27	58000	0
15598044	Female	27	84000	0
15694829	Female	32	150000	1
15600575	Male	25	33000	0
15727311	Female	35	65000	0
15570769	Female	26	80000	0
15606274	Female	26	52000	0
15746139	Male	20	86000	0
15704987	Male	32	18000	0
15628972	Male	18	82000	0
15697686	Male	29	80000	0
15733883	Male	47	25000	1
15617482	Male	45	26000	1
15704583	Male	46	28000	1
15621083	Female	48	29000	1
15649487	Male	45	22000	1
15736760	Female	47	49000	1

Figure 4. Dataset in Logistic Regression

The implementation steps include:

Step 1 .Data preprocessing.

Step 2. Fitting the K-NN algorithm to the training set.

Step 3. Predicting the test results.

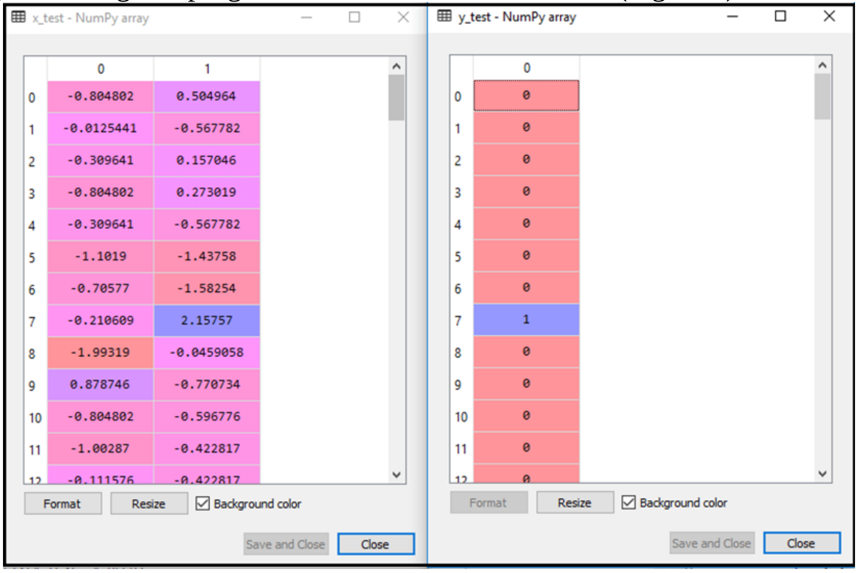
Step 4 .Evaluating test accuracy using a confusion matrix.

Step 5. Visualizing the test set results.

The implementation of the data preprocessing stage is presented in the form of a program listing developed using the Python programming language (Figure 4).

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
data_set= pd.read_csv('user_data.csv')
x= data_set.iloc[:, [11,12]].values
y= data_set.iloc[6].values
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

By executing this program, the test dataset is obtained (Figure 5).



	0	1
0	-0.804802	0.504964
1	-0.0125441	-0.567782
2	-0.309641	0.157046
3	-0.804802	0.273019
4	-0.309641	-0.567782
5	-1.1019	-1.43758
6	-0.70577	-1.58254
7	-0.210609	2.15757
8	-1.99319	-0.0459058
9	0.878746	-0.770734
10	-0.804802	-0.596776
11	-1.00287	-0.422817
12	-0.111576	-0.422817

	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	0
10	0
11	0
12	0

Figure 5. Test Dataset

To apply the K-NN classification algorithm to the training dataset, a Classifier object is created by accessing the KNeighborsClassifier class from the Scikit-learn Neighbors library. Subsequently, the classifier is fitted to the training data using the following code fragment.

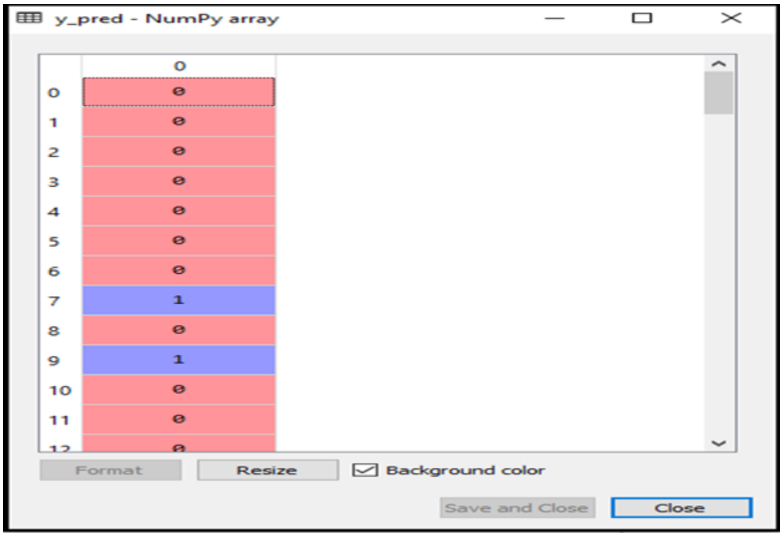
```
from sklearn.neighbors import KNeighborsClassifier
classifier= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=2 )
```

```
classifier.fit(x_train, y_train)
Out[5]:
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=None, n_neighbors=5, p=2,
weights='uniform')
```

To predict the test results, similar to the procedure used in Logistic Regression, a y_pred vector is generated. This is achieved using the following command:

```
y_pred = classifier.predict(x_test).
```

As a result, the y_pred vector containing the predicted class labels is obtained (Figure 5).



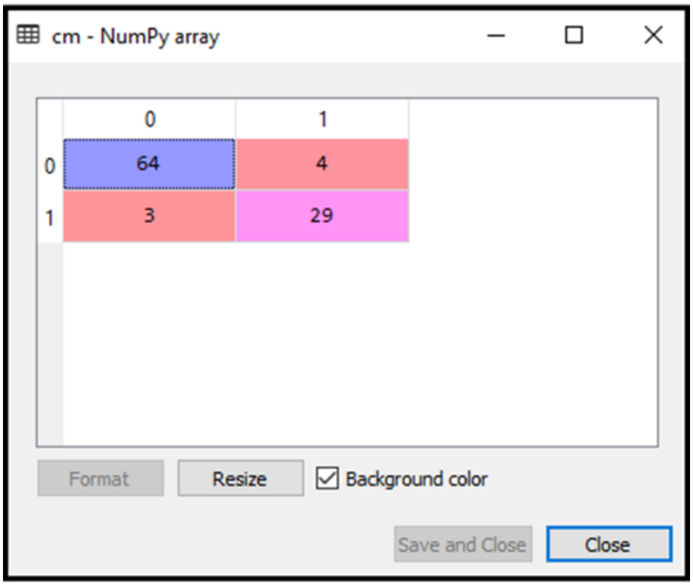
	0
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	1
8	0
9	1
10	0
11	0
12	0

Figure 6. Creating the y_pred vector

The command used to generate the confusion matrix is given as follows:

```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred
```

As a result of its execution, as shown in Figure 6, it can be observed that there are 64 + 29 = 93 correct predictions and 3 + 4 = 7 incorrect predictions, whereas the Logistic Regression model produces 11 incorrect predictions. Therefore, it can be concluded that the model’s performance has been improved through the use of the K-NN algorithm.



	0	1
0	64	4
1	3	29

Figure 7. Generation of the confusion matrix

To visualize the results of the training dataset, the following code fragment is used, which produces the output graph shown in Figure 7:

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('red','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('K-NN Algorithm (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

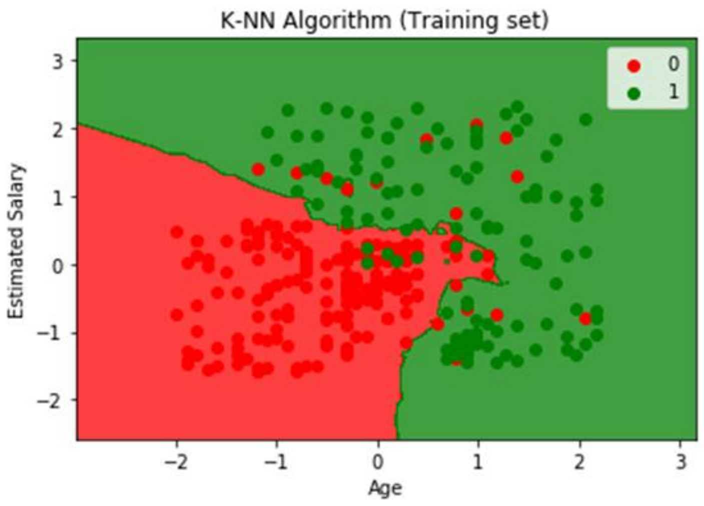


Figure 7. Visualization of the training dataset results

The results demonstrate that the performance of the model improves when using the K-NN algorithm compared to logistic regression, as evidenced by a lower number of incorrect predictions.

3. Conclusion

The experimental results indicate that the K-Nearest Neighbors (K-NN) algorithm outperforms logistic regression in terms of classification accuracy, resulting in a lower number of misclassifications. Furthermore, the visualization of decision boundaries provides clear evidence of the effectiveness of K-NN in modeling non-linear data distributions. Overall, the applied machine learning methods demonstrate strong potential for data analysis and predictive tasks. Moreover, the proposed method enables the effective visualization of large-scale datasets with diverse content.

References

1. Draper, N. R., Smith, H. – Applied Regression Analysis
2. IBM – K-Nearest Neighbors (KNN), https://www.ibm.com/topics/knn
3. James, G., Witten, D., Hastie, T., Tibshirani, R. – An Introduction to Statistical Learning
4. K-Nearest Neighbor (KNN) Algorithm, Last Updated: 23 Dec 2025
5. Khan Academy – Linear Regression
6. Montgomery, D. C., Peck, E. A., Vining, G. G. – Introduction to Linear Regression Analysis
7. Penn State STAT 501 – Regression Methods

8. Python (Scikit-learn Documentation) – Linear Regression
9. Sardarov, Y. B., Ragimov, N. B. – Implementation of a Spam Detector in Python for Corporate Companies, Journal of Electrical Systems, 2024
10. Seber, G. A. F., Lee, A. J. – Linear Regression Analysis
11. Towards Data Science – Machine Learning Basics with the K-Nearest Neighbors Algorithm, https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
12. Wikipedia – K-Nearest Neighbors Algorithm, https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm