Final Project Report: Classification Algorithms in Machine Learning

Mihavana RAHOLDINA FIARA

June 9, 2025

Abstract

This report presents an end-to-end machine learning project focused on classification tasks using real-world weather data. The goal is to preprocess the dataset, explore and visualize important features, apply a variety of classification models, and compare their performance.

Contents

1 Introduction **3**

2 Dataset Overview **4**
2.1 Features Description 5

3 Data Preprocessing **5**

4 Exploratory Data Analysis **5**

5 Modeling Approach **5**
5.1 Linear Regression 5
5.1.1 MAE – Mean Absolute Error 6
5.1.2 MSE - Mean Squared Error 6
5.1.3 R^2 - Coefficient of Determination 6
5.2 Logistic Regression 6
5.3 Decision Tree 6
5.4 K-Nearest Neighbors (KNN) 7
5.5 Support Vector Machines (SVM) 7

6 Evaluation and Results **7**

7 Conclusion **8**

A Appendix: Required libraries **9**

B Appendix: Data preprocessing **9**

C Appendix: Training and Test data **9**

D Appendix: Create, train and predict with Linear regression **9**

E Appendix: Linear Regression Evaluation metrics **10**

F Appendix: Create others models **10**

G Appendix: Train and predict with others models **10**

H Appendix: Calculate values of each metrics **11**

1 Introduction

In this final machine learning project, we explore the application of various supervised learning algorithms to a real-world weather dataset. The primary goal is to predict whether it will rain the next day based on a range of meteorological features such as temperature, humidity, wind speed, and atmospheric pressure. This kind of predictive modeling can be particularly useful for early warning systems and decision-making in weather-sensitive industries.

The project follows a standard machine learning pipeline that includes several key stages: data loading, preprocessing, exploratory data analysis (EDA), model selection, training, and performance evaluation. Throughout the process, we implement and compare several well-known classification algorithms including Linear Regression, Logistic Regression, Decision Tree, K-Nearest Neighbors (KNN), and Support Vector Machines (SVM). Each algorithm is evaluated using appropriate classification metrics to assess accuracy, robustness, and generalization capabilities.

Additionally, we place emphasis on data quality, feature engineering, and the interpretability of results, which are critical in real-world deployment scenarios. By systematically comparing the strengths and limitations of each model, this project aims to identify the most effective approach for this specific prediction task.

2 Dataset Overview

The dataset used in this project originates from the Australian Government's Bureau of Meteorology. It contains daily weather observations from 2008 to 2017. The objective is to predict whether it will rain the next day based on historical features.

Field	**Description**	**Unit**	**Type**
Date	Date of the Observation in YYYY-MM-DD	Date	object
Location	Location of the Observation	Location	object
MinTemp	Minimum temperature	Celsius	float
MaxTemp	Maximum temperature	Celsius	float
Rainfall	Amount of rainfall	Millimeters	float
Evaporation	Amount of evaporation	Millimeters	float
Sunshine	Amount of bright sunshine	Hours	float
WindGustDir	Direction of the strongest gust	Compass Points	object
WindGustSpeed	Speed of the strongest gust	Kilometers/Hour	object
WindDir9am	Wind direction averaged of 10 minutes prior to 9am	Compass Points	object
WindDir3pm	Wind direction averaged of 10 minutes prior to 3pm	Compass Points	object
WindSpeed9am	Wind speed averaged of 10 minutes prior to 9am	Kilometers/Hour	float
WindSpeed3pm	Wind speed averaged of 10 minutes prior to 3pm	Kilometers/Hour	float
Humidity9am	Humidity at 9am	Percent	float
Humidity3pm	Humidity at 3pm	Percent	float
Pressure9am	Atmospheric pressure reduced to mean sea level at 9am	Hectopascal	float
Pressure3pm	Atmospheric pressure reduced to mean sea level at 3pm	Hectopascal	float
Cloud9am	Fraction of the sky obscured by cloud at 9am	Eights	float
Cloud3pm	Fraction of the sky obscured by cloud at 3pm	Eights	float
Temp9am	Temperature at 9am	Celsius	float
Temp3pm	Temperature at 3pm	Celsius	float
RainToday	If there was rain today	Yes/No	object
RISK_MM	Amount of rain tomorrow	Millimeters	float
RainTomorrow	If there is rain tomorrow	Yes/No	float

Table 1: Description of dataset fields

2.1 Features Description

The dataset includes various meteorological features such as temperature, humidity, wind speed, and rainfall statistics. The target variable is `RainTomorrow`, which is a binary classification label.

3 Data Preprocessing

Refer to Appendix B, data preprocessing involved several steps:

- Handling missing values
- Encoding categorical features
- Feature scaling

4 Exploratory Data Analysis

EDA was conducted to better understand the distribution of features and their relationships with the target variable. Plots and summary statistics helped to identify patterns and potential predictors.

5 Modeling Approach

We applied and compared the following classification algorithms:

- Linear Regression
- Logistic Regression
- Decision Tree
- K-Nearest Neighbors (KNN)
- Support Vector Machines (SVM)

5.1 Linear Regression

Linear Regression is a supervised learning algorithm used for predicting a continuous output variable based on one or more input features. It assumes a linear relationship between the input variables and the output. The model finds the best-fit line by minimizing the sum of squared residuals between predicted and actual values.

Example use case: Predicting housing prices based on features like size, location, and number of bedrooms.

Refer to Appendix E, linear regression is evaluated using appropriate metrics which are MAE, MSE and R^2.

5.1.1 MAE – Mean Absolute Error

Mean Absolute Error (MAE) measures the average magnitude of the errors between predicted values and actual values, without considering their direction. It is calculated as the average of the absolute differences between predictions and true values. MAE gives a straightforward interpretation of the average prediction error in the same units as the target variable.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

where y_i is the actual value and $\hat{y}_i$ is the predicted value.

5.1.2 MSE - Mean Squared Error

Mean Squared Error (MSE) calculates the average of the squared differences between predicted and actual values. By squaring the errors, MSE penalizes larger errors more severely than smaller ones. It is commonly used to measure the quality of a regression model and is useful for optimization because it is differentiable.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

5.1.3 R² - Coefficient of Determination

R-squared (R^2) measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It indicates how well the model explains the variability of the outcome data. An R^2 value of 1 means perfect prediction, while 0 means the model does not explain any variability.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

where $\bar{y}$ is the mean of the actual values.

5.2 Logistic Regression

Despite its name, Logistic Regression is used for binary classification tasks. It models the probability that an input belongs to a particular class using the logistic (sigmoid) function. The output is a probability value between 0 and 1.

Example use case: Predicting whether an email is spam or not.

5.3 Decision Tree

A Decision Tree is a non-parametric supervised learning method used for classification and regression. It splits the data into subsets based on feature values, creating a tree-like structure where each internal node represents a test on a feature, and each leaf node represents an output label.

Example use case: Classifying customer churn based on user behavior.

5.4 K-Nearest Neighbors (KNN)

KNN is an instance-based learning algorithm that classifies new data points based on the majority class of their k nearest neighbors in the feature space. It is simple to implement and does not make assumptions about the underlying data distribution.

Example use case: Recommender systems or handwritten digit classification.

5.5 Support Vector Machines (SVM)

SVM is a powerful classification algorithm that works by finding the optimal hyperplane that best separates the data into different classes. It is effective in high-dimensional spaces and can be extended to non-linear classification using kernel functions.

Example use case: Face recognition or text classification.

Each model, except linear regression was trained on a portion of the dataset and evaluated using appropriate metrics such as accuracy, Jaccar index, F1-score and LogLoss (Which is exclusive for Logistic regression). Appendix H

6 Evaluation and Results

The models were evaluated using a test dataset. The results are summarized below:

MAE	MSE	R^2
0.262167	0.118213	0.343821

Table 2: Linear Regression Evaluration Metrics

Model	Accuracy	Jaccar Index	F1-Score	LogLoss
Logistic Regression	0.830534	0.453202	0.623729	6.108161
Decision Tree	0.764885	0.376518	0.547059	
KNN	0.810687	0.354167	0.523077	
SVM	0.755725	0.418182	0.589744	

Table 3: Model Performance Comparison

7 Conclusion

This project demonstrates a complete classification workflow in machine learning, starting from data loading and preprocessing to model training and evaluation. By applying multiple classification algorithms to a real-world weather dataset, we gained practical insights into the behavior and performance of different models under the same conditions.

Among the models tested, **Logistic Regression** emerged as the best-performing approach in terms of both accuracy and robustness, offering a good balance between simplicity and predictive power. Other models like K-Nearest Neighbors and Support Vector Machines also showed competitive performance, although they may require more careful tuning and computational resources.

For future work, several enhancements could be explored to improve model performance. These include hyperparameter optimization using grid search or randomized search, as well as advanced ensemble methods such as Random Forest or Gradient Boosting. Additionally, incorporating feature selection techniques or domain-specific knowledge could further refine the predictive capabilities of the models.

A Appendix: Required libraries

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn import preprocessing
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import Decision
TreeClassifier
from sklearn import svm
from sklearn.metrics import jaccard_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, accuracy_score
import sklearn.metrics as metrics
```

B Appendix: Data preprocessing

```
#convert categorical variables to binary variables
df_sydney_processed = pd.get_dummies(data=df, columns=['RainToday', '
    WindGustDir', 'WindDir9am', 'WindDir3pm'])

#replace the values of the 'RainTomorrow' column changing them from a
    categorical column to a binary column
df_sydney_processed.replace(['No', 'Yes'], [0,1], inplace=True)
```

C Appendix: Training and Test data

```
df_sydney_processed.drop('Date',axis=1,inplace=True)
df_sydney_processed = df_sydney_processed.astype(float)

#Define X (features) and Y
features = df_sydney_processed.drop(columns='RainTomorrow', axis=1)
Y = df_sydney_processed['RainTomorrow']
```

D Appendix: Create, train and predict with Linear regression

```
#Use the `train_test_split` function to split the `features` and `Y`
    dataframes with a `test_size` of `0.2` and the `random_state` set to
    `10`
x_train, x_test, y_train, y_test = train_test_split(features, Y,
    test_size=0.2, random_state=10)
```

```

#Create and train a Linear Regression model called LinearReg using the
    training data (‘x_train‘, ‘y_train‘).
LinearReg = LinearRegression()
LinearReg.fit(x_train,y_train)

#Use the ‘predict‘ method on the testing data (‘x_test‘) and save it to
     the array ‘predictions‘.
predictions = LinearReg.predict(x_test)
```

E Appendix: Linear Regression Evaluation metrics

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
LinearRegression_MAE = mean_absolute_error(y_test, predictions)
LinearRegression_MSE = mean_squared_error(y_test, predictions)
LinearRegression_R2 = np.sqrt(LinearRegression_MSE)
```

F Appendix: Create others models

```
#Create model

#KNN
KNN = KNeighborsClassifier(n_neighbors=4)

#Decision tree
Tree = DecisionTreeClassifier(random_state=101)

#Logistic Regression
LR = LogisticRegression(solver='liblinear')

#SVM
SVM = svm.SVC(class_weight='balanced')
```

G Appendix: Train and predict with others models

```
#model : KNN, Decision Tree, Logistic Regression, SVM

#Training
model.fit(x_train, y_train)

#Prediction
predictions = model.predict(x_test)
```

H Appendix: Calculate values of each metrics

```
#model : KNN, Decision Tree, Logistic Regression(LR), SVM

model_Accuracy_Score = accuracy_score(y_test, predictions)
model_JaccardIndex = jaccard_score(y_test, predictions)
model_F1_Score = f1_score(y_test, predictions)

LR_Log_Loss = log_loss(y_test, predictions)
```