Aalto Yliopisto

CS-C3240 - Machine Learning

Predicting the probability of rain from weather data with regression models

Submitted: December 13, 2023

Contents

1 Introduction: **3**

2 Problem formulation **3**

3 Methods **3**
3.1 Preprocessing of our data set 3
3.2 Logistic Regression Model 4
3.3 Support Vector Machine 4

4 Results **4**
4.1 Logistic regression model 4
4.2 Support Vector Machine 5

5 Conclusion **5**

6 Jupyter Notebook Source Code **6**

1 Introduction:

Weather forecasting relies on analyzing many complex phenomena of the atmosphere. Forecasts are obtained from applying mathematical models on quantitative observation data obtained from the atmosphere, land, and ocean [1].

In this report, we decided to examine whether it would be possible to determine the probability of rain for a given day from other local weather data with a machine learning model. Our project can be classified as supervised machine learning for analyzing a classification problem. For this experiment, we have downloaded weather observation data from Ilmatieteenlaitos from the Helsinki, Kumpula observation point from September 2017 to September 2023 [2]. The data includes measurements such as wind speed, temperature, humidity and air pressure.

In section 2 we will first formulate our problem more formally. In section 3 we'll first discuss how we processed our raw data set to better fit our problem and ML methods. After this, we'll briefly discuss our ML methods and their application for our classification problem. Then in section 4 we examine the results drawn from our ML models and then discuss their applications for actual weather predictions in section 5.

2 Problem formulation

We obtained our data set from Ilmatieteenlaitos' Download observations service. The data set included hourly data of precipitation and other weather measurements. We decided to see if it would be possible to use the other weather measurements to correctly predict rainfall.

Our first step was to decide what metric we'd use to determine rainfall. We ended up deciding that a binary label for a given day being rainy or not would suit our data set the best. A common metric for a given day being classified as rainy is if its total precipitation is more than 1 mm [3]. Let's note for this paper $y = 1$ for a rainy day and $y = 0$ for a non-rainy day. Then the other hourly weather measurements with continuous data would define our feature matrix, which determines if the day is rainy or not in our classification problem.

3 Methods

3.1 Preprocessing of our data set

The raw data set obtained from Ilmatieteenlaitos includes hourly weather observations from September 17th, 2017 to September 17th, 2023 resulting in a total of around 52 000 raw data points. To reduce the computational requirements of our models, we first reduced our data set to a more manageable size. We achieved this by first reducing our data points to hourly measurements of temperature, humidity, wind speed, precipitation, and air pressure, which we determined to have the highest correlation with precipitation, and discarding other observations. We then formed four features from the hourly data: the daily average temperature, wind speed, humidity, and air pressure. We then calculated the daily average precipitation and classified each day with a binary label according to the condition mentioned in section 2.

We also noticed our raw data was missing values for some measurements. We decided to fill these gaps with forward fill (i.e. by copying the previous hour's data) as we reasoned that weather data is unlikely to change majorly during one hour and thus forward fill's

effects on the daily averages would be minor.

We then split our processed data into a training, validation and test set. The training data was decided to be 60% of our processed data points as a larger number of data points would increase our logistic model's accuracy. We also required enough data to sufficiently validate our model so 20% of the processed data was allocated for the validation set and the remaining 20% of data for the test set to test how efficiently our model works. Each data point from the processed 2 200 data points was randomly picked into one of the sets. We will use this split data for the training and testing of both of our ML models.

3.2 Logistic Regression Model

We decided to first try to predict rainfall with logistic regression model. Logistic regression is often a good choice for binary classification problems, so it seemed to be a good fit four our problem. With logistic regression our problem could be formulated as:

$$h(X) \geq 0.5 \tag{1}$$

if a given day is rainy ($y = 1$), and

$$h(X) < 0.5 \tag{2}$$

if the day is not rainy ($y = 0$). Here $h(X) = \frac{1}{1+e^{-\mathbf{wX}}}$ is a logistic regression function, where $\mathbf{w}$ is a vector, which consists of weight parameters and $\mathbf{X}$ is the modified data points [4].

In our logistic regression model, we decided to use the logistic loss function since it is continuous for all values of the features [4] is commonly used with logistic regression models and suits a binary classification problem.

3.3 Support Vector Machine

For our second regression model we decided to use a support vector machine with C-support (SVC), where the idea is to find a linear hyperplane in our N-dimensional space (N = the number of features), which separates the data distinctly in to two subsets, where one corresponds a non-rainy day ($y = 0$) and a rainy day ($y = 1$) [5] [6] . In brief, the task in SVM is to find the optimal hyper plane, which separates the data with maximum marginal for the hyperplane [4]. A SVM was a good fit for our weather data because it is usually used with multidimensional data set and is applicable for binary classification problems [7].

In this model we used a hinge loss function, which is common in SVM because it is very efficient in classification problems [4].

4 Results

4.1 Logistic regression model

From the logistic regression's confusion matrix in figure 1 we can see that our model has a very high accuracy for true negatives where as its accuracy for true positives is lower. This is most likely influenced by the relatively low amount of positive labels in our data set. In combination with the accuracy score of 80% we can thus deduce that our model has a relatively high likelihood of correctly predicting a non-rainy day correctly but a lower likelihood of correctly predicting a rainy day.

However, our logistic regression model error for correct predictions is rather high with the training error being 6.9, validation error being 7.1, and finally the test error being 6.0. Thus our model's error for the predictions it makes is rather high.

4.2 Support Vector Machine

Similarly from our SVM's confusion matrix in figure 1 we can see that our model has a very high accuracy for true negatives where as its accuracy for true positives is lower with the total accuracy of 83% being slightly higher than our logistic regression model's.

Our SVM's errors are significantly lower than our logistic regression model's with the training error being 0.85, the validation error being 0.89 and finally the test error being 0.79. This implies that our SVM predicts rainy days much more efficiently than our logistic regression model and should thus be chosen as the final method for predicting precipitation.

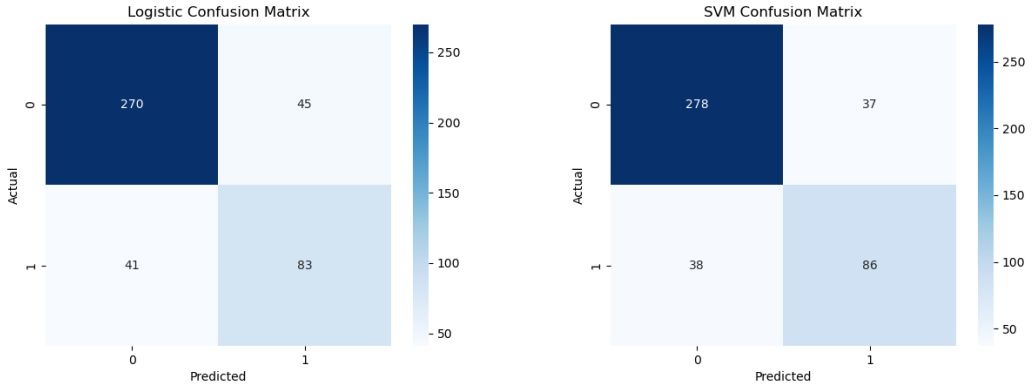


Figure 1: Confusion matrices for the logistic regression and SVM

5 Conclusion

In conclusion, the support vector machine model is better classifying days was rainy from weather data. This is based on the accuracy scores, as well as the training, validation, and test errors for both models used in this paper. The model still has a relatively low probability of correctly classifying a rainy day where as its probability for correctly classifying a non-rainy day is much higher.

The reason for the SVM's lowe error in predicting the weather, is most likely that the SVM intended to optimize the classification problem with higher dimension of features where as logistic regression is mostly intended to be used when we compare only a singular feature with a label. Therefore, the logistic model is not efficient to use when analyzing multi-dimensional data.

However, the difference in ML methods accuracy's were small. To order see which method is better for predicting a rainy day, we should have more data or we should have drop some features, which have poor correlation with the label. More research into the variables' in-between correlations should also be made. In addition, acknowledging the effects of different seasons would most likely improve our model.

As it is, our model could be used for correctly predicting whether a given day is most likely not going to be rainy if predictions about the other measurements can be correctly made. By training our model with some modifications made to the data set used in the problem to account, for example accounting for the previous days' weather history and creating new features from the history, possible predictions about the probability of rain could also be made only from weather observations without requiring other predictions of weather data.

References

[1] The Conversation, The science of weather forecasting: what it takes and why it's so hard to get right , Published 01.02.2022, Read 20.09.2023

[2] Weather data, 17.09.2017 - 17.09.2023,: Ilmatieteenlaitos, Observation point: Helsinki, Kumpula, data downloaded 18.09.2023 12:00

[3] Rainfall Classification: Intensity of Rainfall in 24 Hours, Weather and Climate Services Division, NCHM

[4] Machine Learning - Regression Aalto University, published 8.9.2023, Read 20.09.2023

[5] Diving into C-Support Vector Classification, Gustavo Santos, Published Oct 18, 2022, Read Oct 10, 2023.

[6] Support Vector Machine — Introduction to Machine Learning Algorithms, Rohith Gandhi, Published Jun 7, 2018, Read oct 10, 2023

[7] SVM Binary Classification Grid Gain documentation, Read Oct 11, 2023

6 Jupyter Notebook Source Code

Source code

October 11, 2023

[47]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import datasets
from sklearn.tree import export_text, plot_tree, DecisionTreeClassifier,␣
 ↪export_graphviz
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, roc_curve, auc,␣
 ↪precision_recall_curve
from sklearn.metrics import  average_precision_score, classification_report,␣
 ↪accuracy_score, log_loss, hinge_loss
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.decomposition import PCA
from sklearn.svm import SVC
```

[48]:
```
df1 = pd.read_csv("Data2017-2020 .csv")
df2 = pd.read_csv("Data2020-2023.csv")
#Concating our two data frames
data = pd.concat([df1,df2])
df4 = pd.concat([df1,df2])
n_rawDataPoints = data.shape[0] #The amount of raw data points
#Creating a single "datetime" column from the data's year, month and day data.
data = data.assign(datetime = data["Vuosi"].astype(str)+'-'+data["Kk"].
 ↪astype(str)+'-'+data["Pv"].astype(str))
#Excludint the unnecessary columns from the data set.
data = data.drop(['Aikavyöhyke','Vuosi','Kk','Pv','Klo'],axis=1)
data['datetime'] = pd.to_datetime(data['datetime'])

data = data.rename(columns={   #Renaming all of the columns to English
                           'Ilmanpaine (msl) (hPa)' :'Air Pressure (hPa)',
                           'Sademäärä (mm)':'Precipitation (mm)',
                           'Suhteellinen kosteus (%)': 'Humidity (%)',
                           'Tuulen nopeus (m/s)' : 'Wind speed (m/s)',
```

```
                            'Ilman lämpötila (degC)' : 'Temperature (degC)' })

#Forcing all of the datapoints to numeric values
numeric_columns = [col for col in data.columns if col != 'datetime']
for column in numeric_columns:
    data[column] = pd.to_numeric(data[column], errors='coerce')
#We'll notice that the data contains many NaNs.
#We can replace NaNs from the hourly data with forward fill:
data.fillna(method='ffill', inplace=True)

#Grouping all of the data by the "datetime" column's values
#Then calculating the daily average values for the feature values.
data_avrg = data.groupby(data['datetime']).mean().reset_index()
n_DataPoints = data_avrg.shape[0]
#Creating the binary data for a given day being rainy or not
data_avrg['Total precipitation (mm)'] = data_avrg['Precipitation (mm)'] * 24
data_avrg['Is rainy'] = (data_avrg['Total precipitation (mm)'] >= 1).astype(int)

n_rainyDays = sum(data_avrg['Is rainy'])
```

[49]:
```
df3 = data_avrg[['Air Pressure (hPa)', 'Humidity (%)', 'Is rainy','Wind speed␣
 ↪(m/s)', 'Temperature (degC)']]
df3 = df3.rename(columns={  'Is rainy' : 'IR',
                            'Air Pressure (hPa)' : 'AP',
                            'Humidity (%)': 'H',
                            'Wind speed (m/s)' : 'WS',
                            'Temperature (degC)' : 'T'})
matrix = df3.corr()
print(matrix)
```

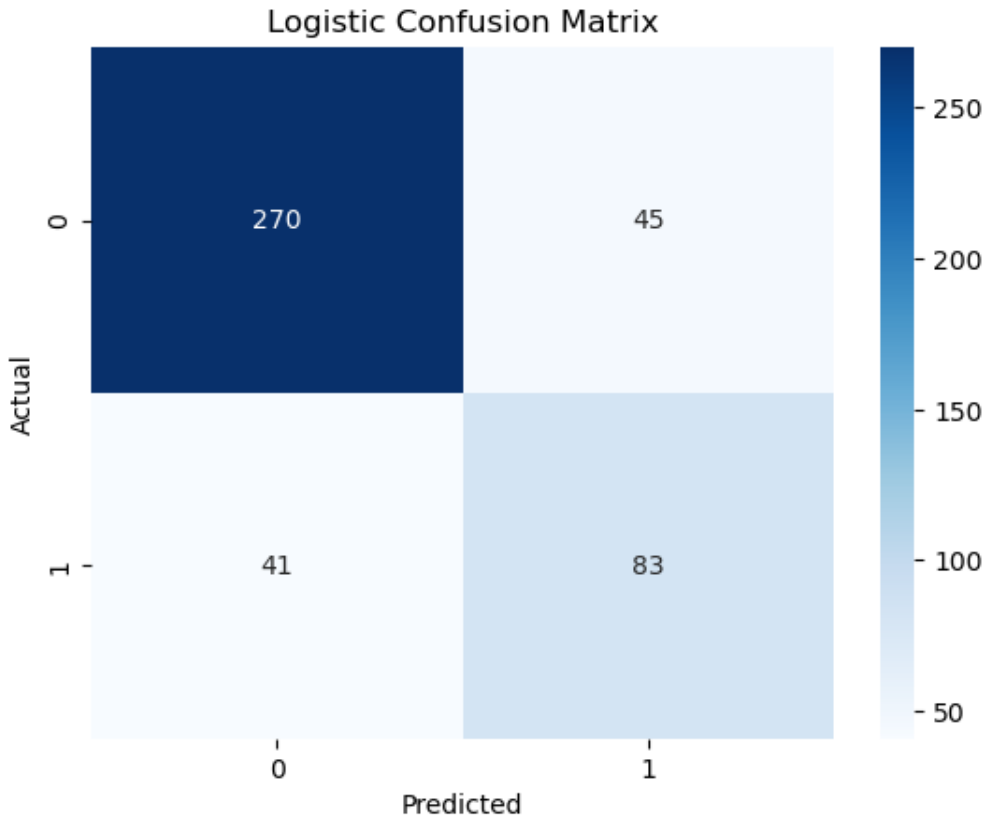
```
          AP         H        IR        WS         T
AP  1.000000 -0.346764 -0.456409 -0.316105  0.094665
H  -0.346764  1.000000  0.495886  0.148782 -0.341849
IR -0.456409  0.495886  1.000000  0.278073 -0.115520
WS -0.316105  0.148782  0.278073  1.000000 -0.110512
T   0.094665 -0.341849 -0.115520 -0.110512  1.000000
```

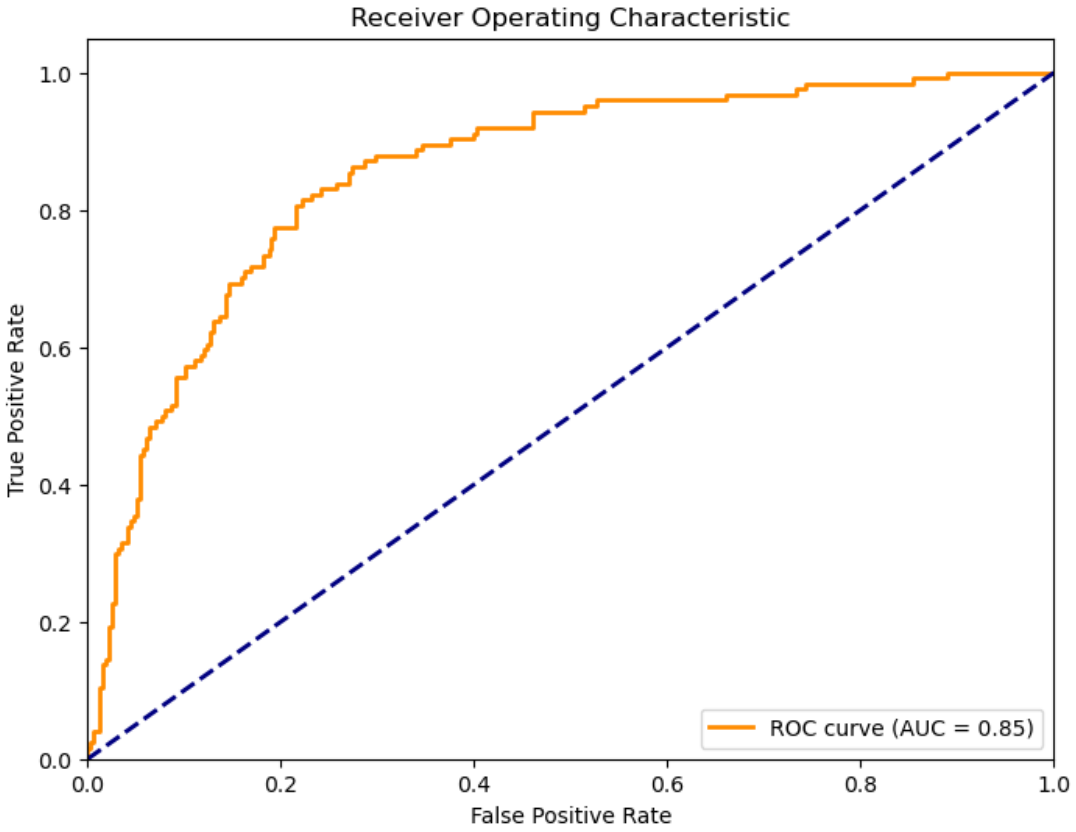
[50]:
```
y = data_avrg['Is rainy']
X = data_avrg[['Air Pressure (hPa)', 'Humidity (%)','Wind speed (m/s)',␣
 ↪'Temperature (degC)']]

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.4,␣
 ↪random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,␣
 ↪random_state=42)

clf = LogisticRegression(solver='liblinear')
```

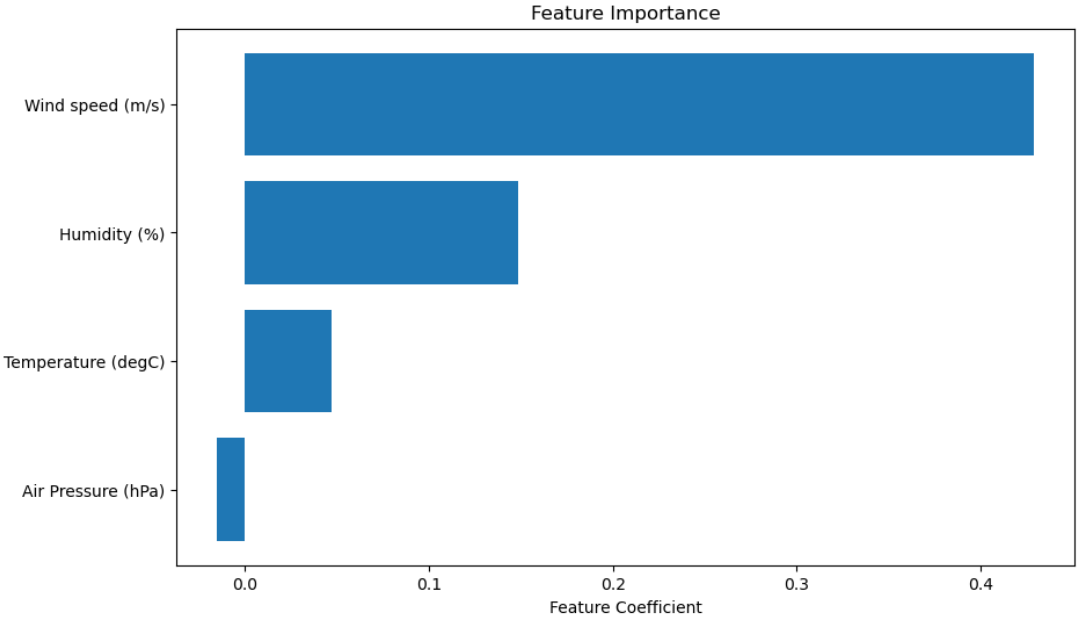
```
clf.fit(X_train,y_train)

y_pred_train = clf.predict(X_train)

tr_log_loss = log_loss(y_train,y_pred_train)

y_pred_val =clf.predict(X_val)

val_log_loss = log_loss(y_val,y_pred_val)

y_pred_test = clf.predict(X_test)

test_log_loss = log_loss(y_test, y_pred_test)

accuracy_Logistic = accuracy_score(y_val, y_pred_val)

# Confusion Matrix

conf_matrix = confusion_matrix(y_val, y_pred_val)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Logistic Confusion Matrix')
plt.savefig("LogisticConfusionMatrix.png")

# ROC Curve
y_prob = clf.predict_proba(X_val)[:, 1]
fpr, tpr, thresholds = roc_curve(y_val, y_prob)
roc_auc = auc(fpr, tpr)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.
  ↪2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc='lower right')
plt.savefig("LogisticROC.png") # save as png

# Feature Importance
# Sort features by their coefficients
coef = clf.coef_[0]
feature_names = X.columns
coef_indices = np.argsort(coef)
```

```
# Create a bar plot with sorted features
plt.figure(figsize=(10, 6))
plt.barh([feature_names[i] for i in coef_indices], coef[coef_indices],␣
 ↪align='center')
plt.xlabel('Feature Coefficient')
plt.title('Feature Importance')
plt.savefig("LogisticFeatureImportance.png") # save as png

print(accuracy_Logistic)
```

0.8041002277904328



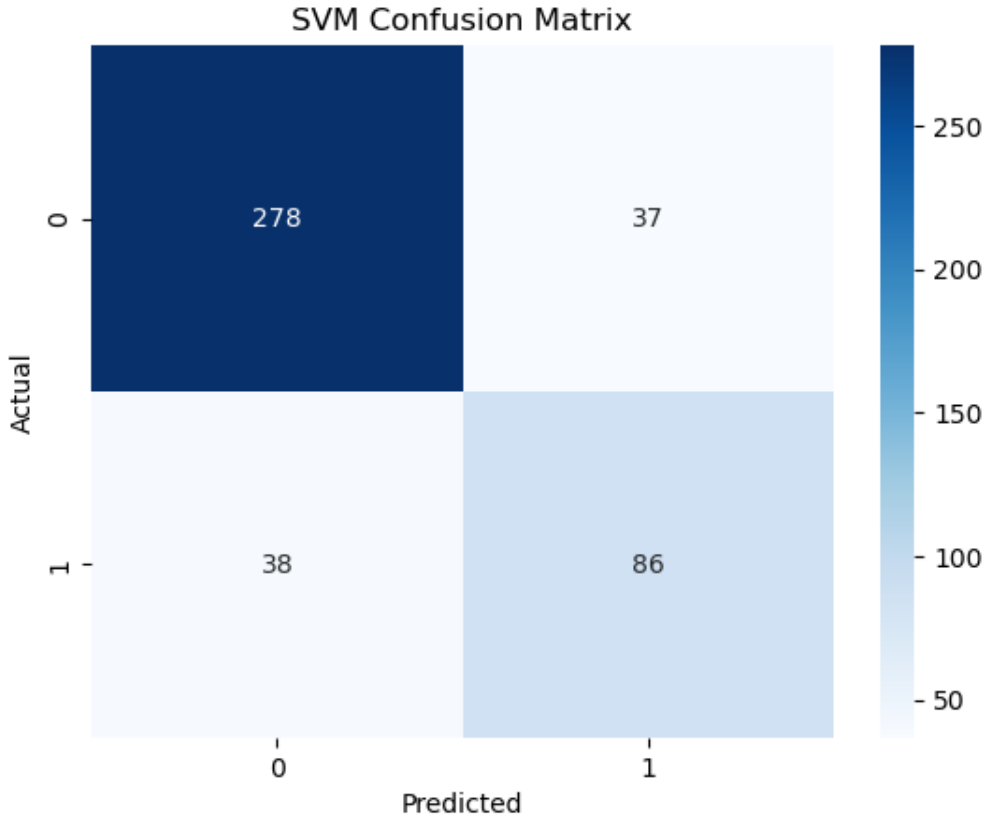

Receiver Operating Characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (AUC = 0.85)

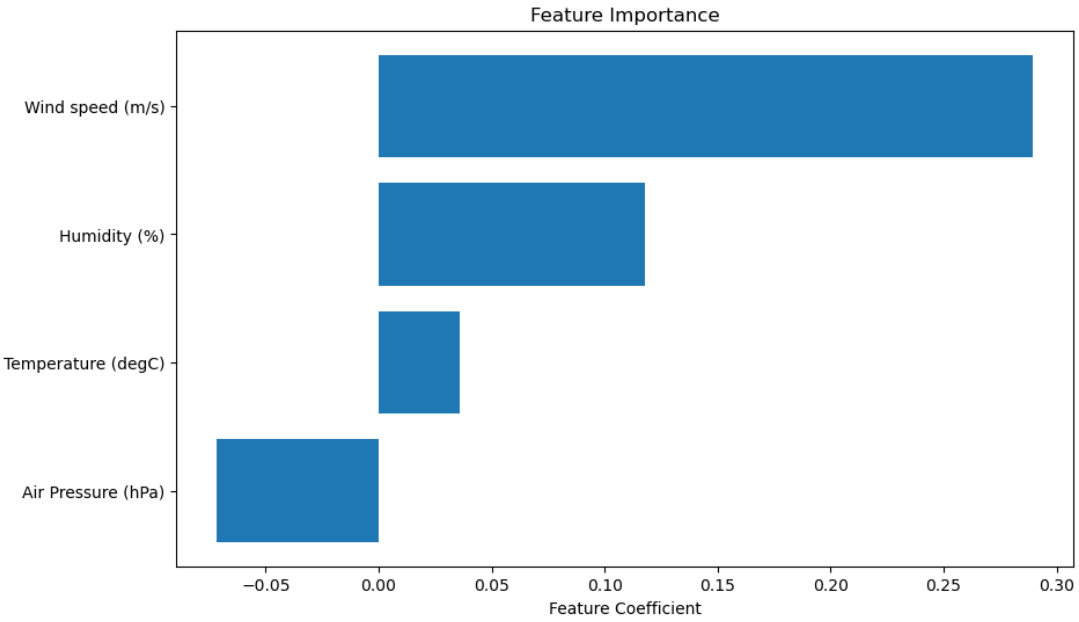

Feature Importance
Wind speed (m/s)
Humidity (%)
Temperature (degC)
Air Pressure (hPa)
0.0
0.1
0.2
0.3
0.4
Feature Coefficient

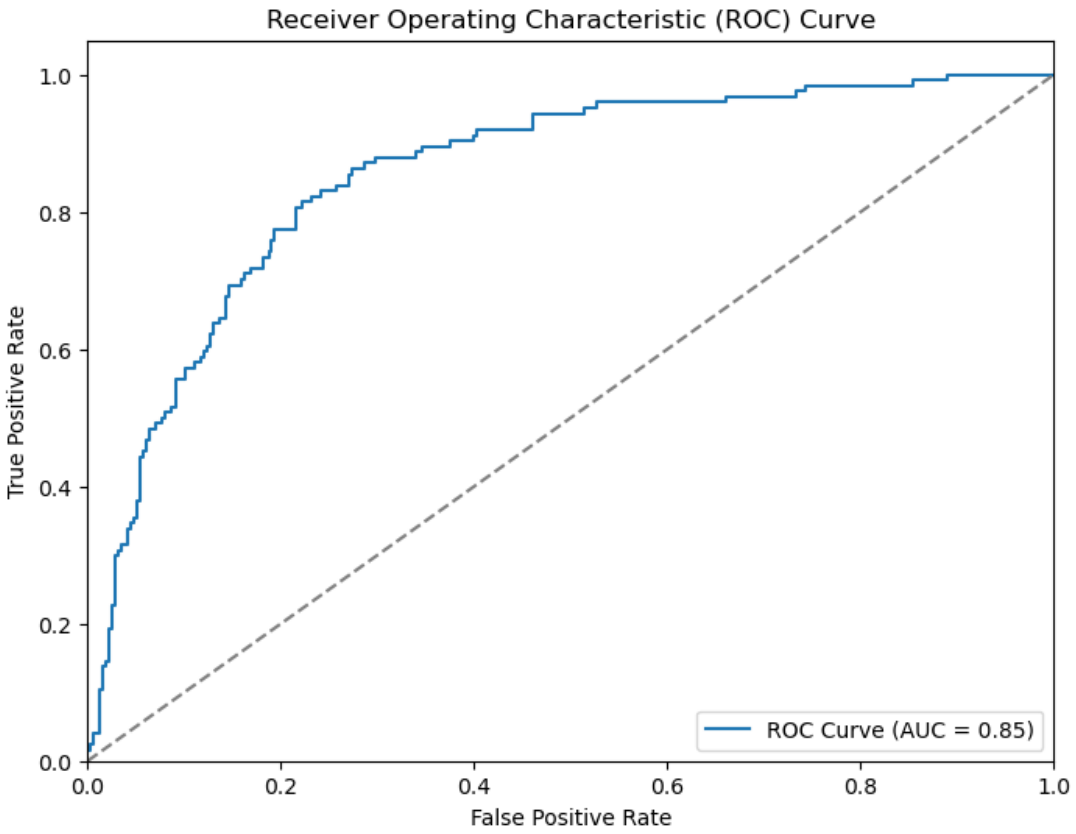
[51]:
```
clf = SVC(kernel = 'linear', C =1.0)
clf.fit(X_train, y_train)


y_pred_train = clf.predict(X_train)

tr_hinge_loss = hinge_loss(y_train,y_pred_train)

y_pred_val = clf.predict(X_val)

val_hinge_loss = hinge_loss(y_val,y_pred_val)

y_pred_test = clf.predict(X_test)

test_hinge_loss = hinge_loss(y_test, y_pred_test)

conf_matrix = confusion_matrix(y_val, y_pred_val)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('SVM Confusion Matrix')
plt.savefig("SVCConfusionMatrix.png") # save as png


print(accuracy_score(y_val,y_pred_val))
# Feature Importance
# Sort features by their coefficients
coef = clf.coef_[0]
feature_names = X.columns
coef_indices = np.argsort(coef)
plt.show()

# Create a bar plot with sorted features
plt.figure(figsize=(10, 6))
plt.barh([feature_names[i] for i in coef_indices], coef[coef_indices],␣
 ↪align='center')
plt.xlabel('Feature Coefficient')
plt.title('Feature Importance')
plt.savefig("SVCFeatureImportance.png") # save as png


plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
```

```
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.savefig("SVC-ROC.png") # save as png
```

0.8291571753986332




Feature Importance
Wind speed (m/s)
Humidity (%)
Temperature (degC)
Air Pressure (hPa)
−0.05
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Feature Coefficient


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC Curve (AUC = 0.85)

[52]:
```
print(f"Thus we have training, validation, and test losses for the logistic␣
↪regression: {tr_log_loss:.2f}, {val_log_loss:.2f}, {test_log_loss:.2f}\n and␣
↪training, validation, and test losses for the SVM: {tr_hinge_loss:.2f},␣
↪{val_hinge_loss:.2f}, {test_hinge_loss:.2f}")
```

```
Thus we have training, validation, and test losses for the logistic regression:
6.93, 7.06, 5.99
 and training, validation, and test losses for the SVM: 0.85, 0.89, 0.79
```