WEATHER PREDICTION

ANNA JACOBSON | EILEEN MCCONVILLE | GOVIND KISHAN | ALAN ROCQUE | AKIL RAMMOHAN

Our Topic

- Weather Patterns
  - Can we predict whether or not it will rain on a given day with given weather factors?
- Why is it important?
  - What we wear
  - Activities we plan
  - Methods of Travel
- Our statistical methods
  - KNN Regression
  - Logistic Regression

Data

- Data read in from Kaggle
- Our "X" columns/features
  - StandardScaler to normalize the features
- Our "y" column: rain
  - One hot encoding for categorical variable

	Temperature	Humidity	Wind_Speed	Cloud_Cover	Pressure	Rain
0	23.720338	89.592641	7.335604	50.501694	1032.378759	rain
1	27.879734	46.489704	5.952484	4.990053	992.614190	no rain
2	25.069084	83.072843	1.371992	14.855784	1007.231620	no rain
3	23.622080	74.367758	7.050551	67.255282	982.632013	rain
4	20.591370	96.858822	4.643921	47.676444	980.825142	no rain
...	...	...	...	...	...	...
2495	21.791602	45.270902	11.807192	55.044682	1017.686181	no rain
2496	27.558479	46.481744	10.884915	39.715133	1008.590961	no rain
2497	28.108274	43.817178	2.897128	75.842952	999.119187	no rain
2498	14.789275	57.908105	2.374717	2.378743	1046.501875	no rain
2499	26.554356	97.101517	18.563084	81.357508	1001.729176	no rain

2500 rows × 6 columns

Data (Modified)

Temperature	Humidity	Wind_Speed	Cloud_Cover	Pressure	Rain	rain
23.720338	89.592641	7.335604	50.501694	1032.378759	rain	1
27.879734	46.489704	5.952484	4.990053	992.614190	no rain	0
25.069084	83.072843	1.371992	14.855784	1007.231620	no rain	0
23.622080	74.367758	7.050551	67.255282	982.632013	rain	1
20.591370	96.858822	4.643921	47.676444	980.825142	no rain	0
...	...	...	...	...	...	...
21.791602	45.270902	11.807192	55.044682	1017.686181	no rain	0
27.558479	46.481744	10.884915	39.715133	1008.590961	no rain	0
28.108274	43.817178	2.897128	75.842952	999.119187	no rain	0
14.789275	57.908105	2.374717	2.378743	1046.501875	no rain	0
26.554356	97.101517	18.563084	81.357508	1001.729176	no rain	0

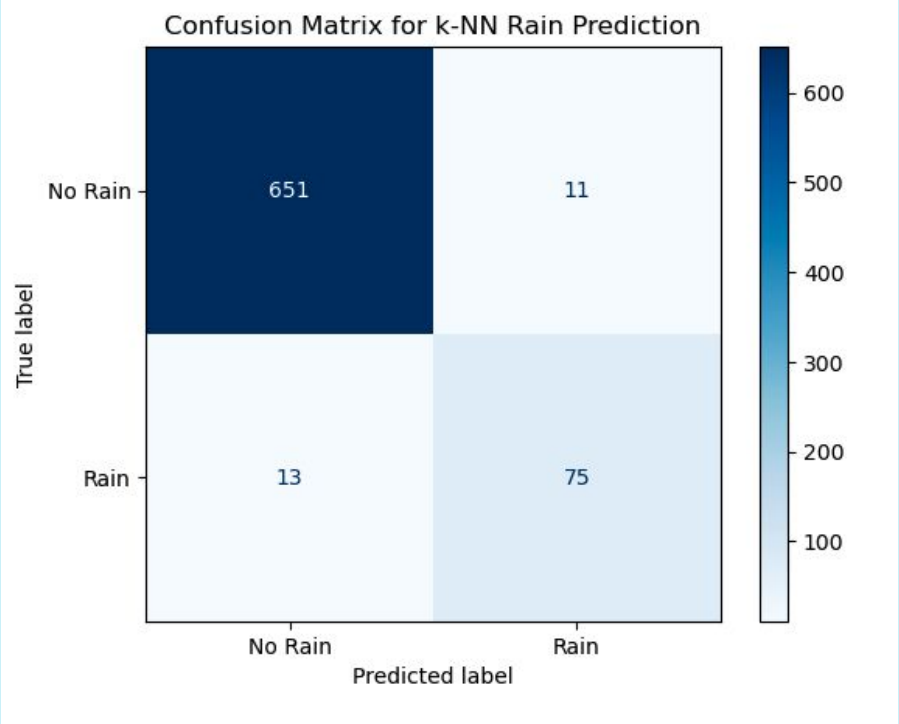
X_scaled

```
array([[ 0.1554307 ,  1.26539348, -0.44481385,  0.02897213,  0.8947143 ],
       [ 0.72322527, -0.89507374, -0.6841429 , -1.53407371, -1.07457032],
       [ 0.33954662,  0.93859882, -1.47673073, -1.19524633, -0.35066253],
       ...,
       [ 0.75442305, -1.0290299 , -1.21282806,  0.89928899, -0.75241945],
       [-1.0637386 , -0.32274425, -1.3032236 , -1.6237562 ,  1.59414185],
       [ 0.54229936,  1.64176416,  1.49793905,  1.08868017, -0.62316338]])
```

Model #1: KNN Code

```python
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report

# Load the dataset
file_path = 'weather_forecast_data.csv'  # Replace with your file path
weather_data = pd.read_csv(file_path)

# Preprocessing the data
# Map 'Rain' column to numerical values (1 for rain, 0 for no rain)
weather_data['Rain'] = weather_data['Rain'].map({'rain': 1, 'no rain': 0})

# Separate features and target variable
X = weather_data.drop(columns='Rain')
y = weather_data['Rain']

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)

# Initialize and train the k-NN classifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Predict on the test data
y_pred = knn.predict(X_test)

# Generate and print the classification report
report = classification_report(y_test, y_pred, target_names=['No Rain', 'Rain'])
print(report)
```

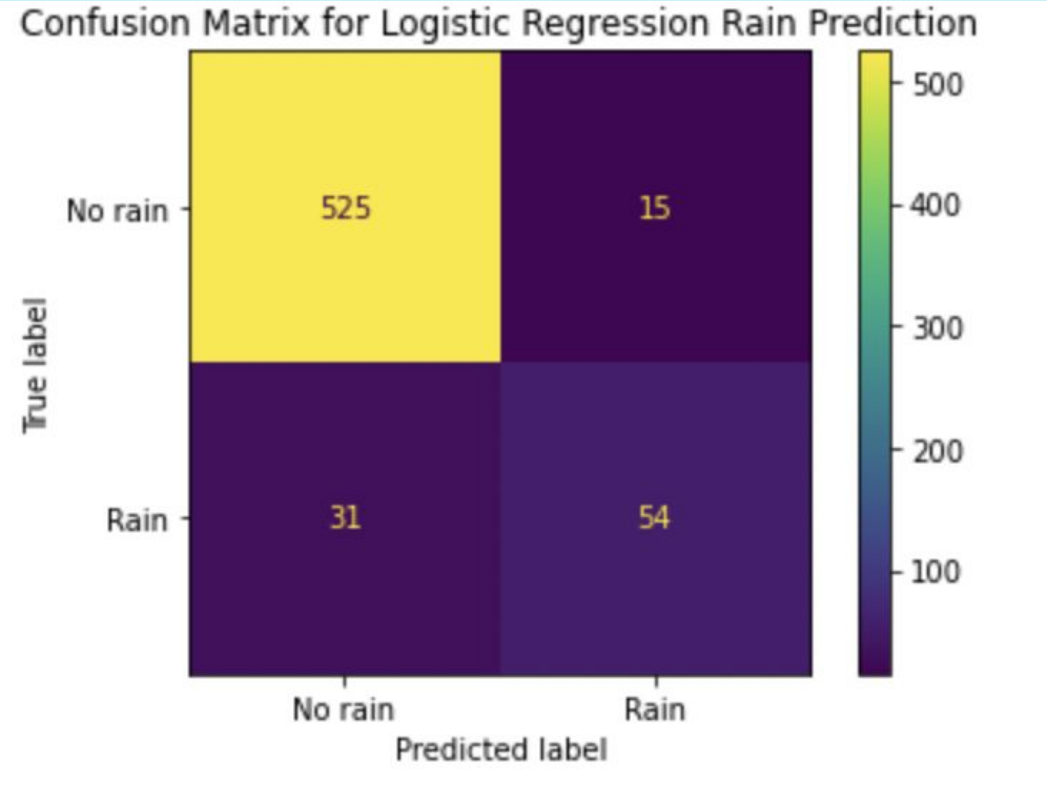
Model 1: KNN Model



Model #2: Logistic Regression Code

```
from matplotlib import pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score
import pandas as pd
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


X = m[['Temperature', 'Humidity', 'Wind_Speed', 'Cloud_Cover', 'Pressure']]
y = m['rain']
scaler = StandardScaler()
X_reduced = scaler.fit_transform(X)
X_train,X_test,y_train,y_test= train_test_split(X_reduced,y,test_size=0.25,random_state=0)
logistic_regression= LogisticRegression()
logistic_regression.fit(X_train,y_train)
y_pred=logistic_regression.predict(X_test)
ConfusionMatrixDisplay.from_estimator(logistic_regression, X_test, y_test)
plt.title('Confusion Matrix for Logistic Regression Rain Prediction')
plt.show()
precision = precision_score(y_test, y_pred)
print(f"Precision:{precision:.3}")
```

Model #2: Logistic Regression



Outcome of Models

- Measuring on precision
- False positives
- KNN is the best model for prediction due to the precision scores
  - 92.6% for the overall model
- Using KNN we can predict rain or no rain accurately 92.6% of the time