ForestFires

April 17, 2025

[38]:

```python
# Imports
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.cluster import KMeans
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
```

[52]:

```python
# Load dataset
df = pd.read_csv("./weatherHistory.csv")
```

[53]:

```python
# Preprocessing
df_clean = df[['Temperature (C)', 'Humidity', 'Wind Speed (km/h)', 'Wind␣
  ↪Bearing (degrees)', 'Summary']].dropna()
np.random.seed(42)
df_clean['Hot Spot Count'] = np.random.randint(1, 20, size=len(df_clean))
df_clean['Confidence Level'] = np.random.randint(1, 4, size=len(df_clean))
le = LabelEncoder()
df_clean['Weather Category'] = le.fit_transform(df_clean['Summary'])
df_clean.drop(columns='Summary', inplace=True)
```

[54]:

```python
# Normalize features
scaler = MinMaxScaler()
scaled_features = scaler.fit_transform(df_clean)
df_scaled = pd.DataFrame(scaled_features, columns=df_clean.columns)
```

[55]:

```python
# Clustering
kmeans = KMeans(n_clusters=5, random_state=42, n_init=10)
df_scaled['Risk Level'] = kmeans.fit_predict(df_scaled)
```

[56]:

```python
# Classification
X = df_scaled.drop(columns=['Risk Level'])
y = df_scaled['Risk Level']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
conf_matrix = confusion_matrix(y_test, y_pred)
```

[57]:

```
# Classification report
classification_rep = classification_report(y_test, y_pred, output_dict=True)
classification_df = pd.DataFrame(classification_rep).transpose()
print("\n  Classification Report:")
print(classification_df.round(2))
```

```

  Classification Report:
              precision  recall  f1-score  support
0                  0.99    0.99      0.99   6188.0
1                  1.00    1.00      1.00   4930.0
2                  1.00    1.00      1.00   6581.0
3                  1.00    1.00      1.00   6510.0
4                  1.00    1.00      1.00   4727.0
accuracy           1.00    1.00      1.00      1.0
macro avg          1.00    1.00      1.00  28936.0
weighted avg       1.00    1.00      1.00  28936.0
```

[58]:

```
# Final Risk Analysis by Cluster
risk_analysis = df_clean.copy()
risk_analysis['Risk Level'] = df_scaled['Risk Level']
```

[59]:

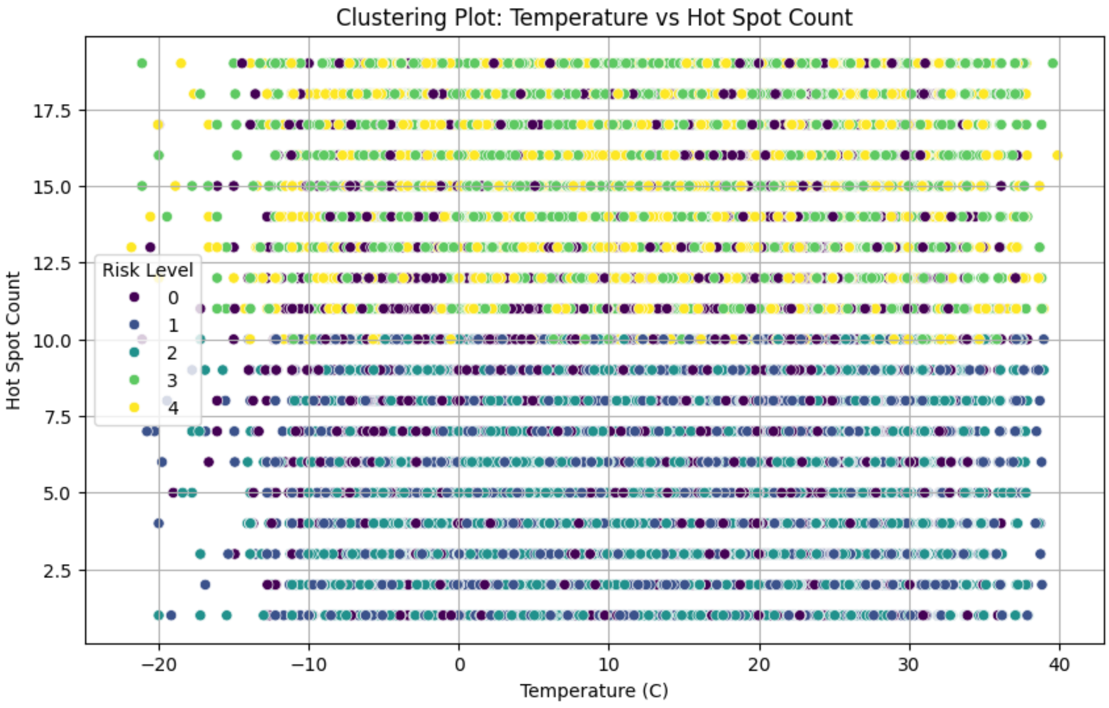
```
# Group by cluster and compute mean conditions
risk_summary = risk_analysis.groupby('Risk Level').mean()[[
    'Temperature (C)',
    'Humidity',
    'Wind Speed (km/h)',
    'Wind Bearing (degrees)',
    'Hot Spot Count',
    'Confidence Level'
]]

print("\n Final Risk Analysis by Risk Level (Cluster Centroids):")
print(risk_summary.round(2))
```

```

 Final Risk Analysis by Risk Level (Cluster Centroids):
            Temperature (C)  Humidity  Wind Speed (km/h)  \
Risk Level
0                     11.64      0.73               9.92
1                     12.01      0.73              10.93
```

```
2                     12.06      0.74             11.20
3                     12.02      0.74             11.18
4                     11.93      0.74             10.79

            Wind Bearing (degrees)  Hot Spot Count  Confidence Level
Risk Level
0                            63.94           10.08              2.50
1                           193.47            5.37              1.00
2                           243.33            4.90              2.47
3                           245.12           15.08              2.53
4                           182.98           14.87              1.00
```

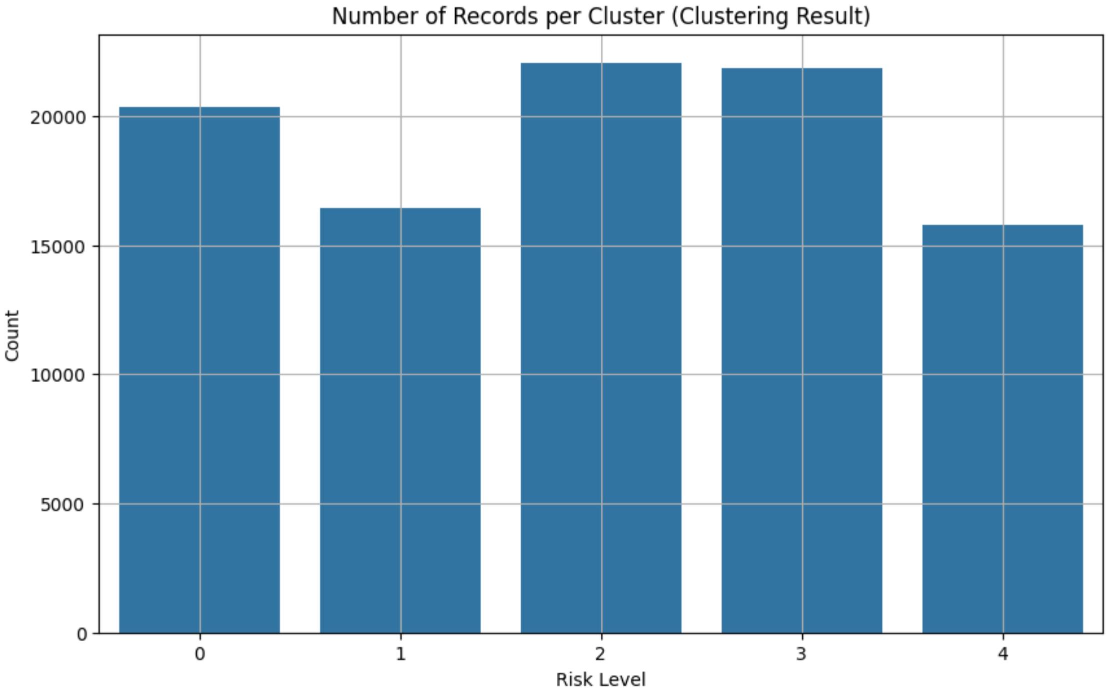
[60]:
```python
# Restore original values for plotting
df_plot = df_scaled.copy()
df_plot['Temperature'] = df_clean['Temperature (C)'].values
df_plot['Hot Spot Count'] = df_clean['Hot Spot Count'].values
```

[61]:
```python
# Temperature vs Hot Spot Count
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df_plot, x='Temperature', y='Hot Spot Count', hue='Risk␣
 ↪Level', palette='viridis')
plt.title('Clustering Plot: Temperature vs Hot Spot Count')
plt.xlabel('Temperature (C)')
plt.ylabel('Hot Spot Count')
plt.grid(True)
plt.show()
```



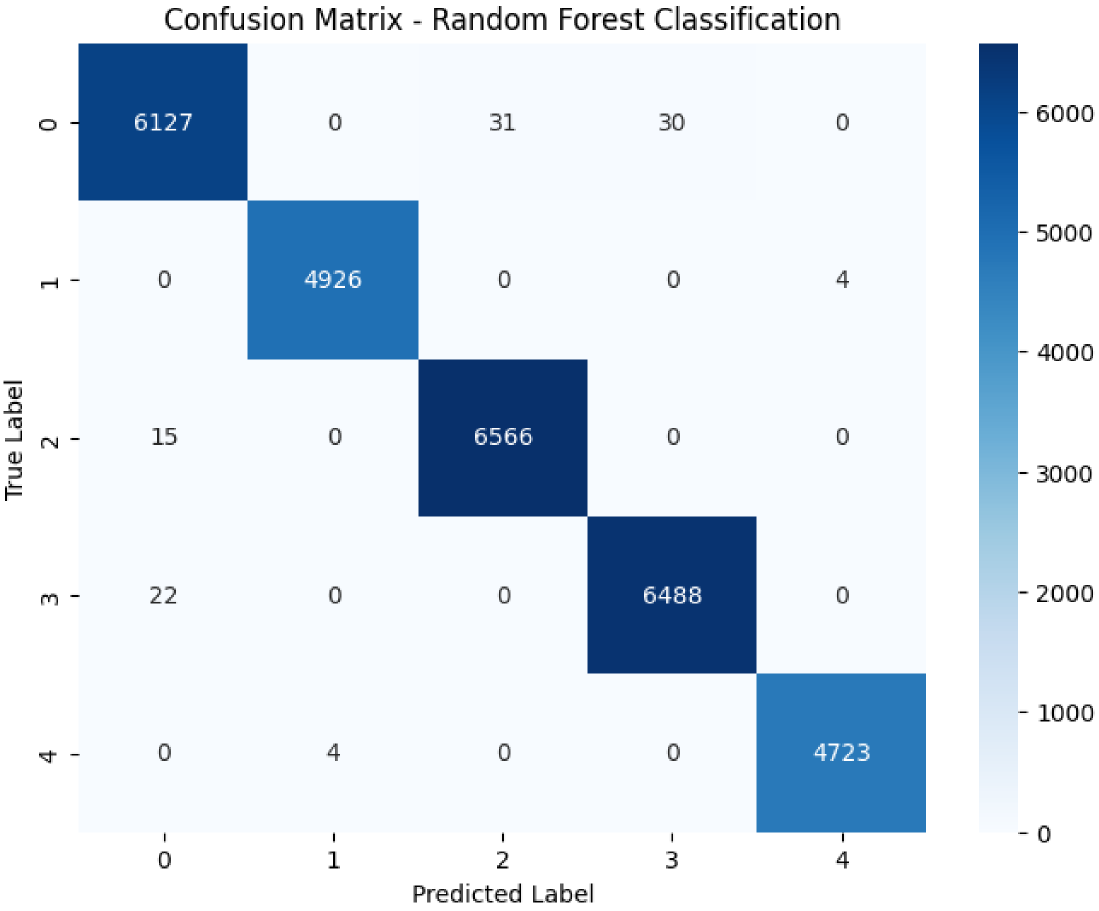
[62]:

```
# Clustering Result Distribution
plt.figure(figsize=(10, 6))
sns.countplot(x='Risk Level', data=df_plot)
plt.title('Number of Records per Cluster (Clustering Result)')
plt.xlabel('Risk Level')
plt.ylabel('Count')
plt.grid(True)
plt.show()
```



[63]:

```
# Confusion Matrix for Classification
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',␣
 ↪xticklabels=range(5), yticklabels=range(5))
plt.title('Confusion Matrix - Random Forest Classification')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



[64]:

```
# Save the trained model
model_path = "./random_forest_forest_fire_model.pkl"
joblib.dump(rf, model_path)

model_path
```

[64]:

```
'./random_forest_forest_fire_model.pkl'
```