**Kubernetes Failure Prediction**

Project Documentation

**1. Introduction**

Kubernetes is a widely adopted platform for deploying and managing containerized applications. However, failures in Kubernetes clusters can significantly impact application availability and performance. This project aims to build a machine learning model using a Random Forest Classifier to predict potential failures in a Kubernetes cluster. By analyzing system metrics and event logs, the model helps improve cluster reliability and minimize downtime.

**2. Objectives**

- Monitor and analyze Kubernetes failures.

- Train a machine learning model to predict failures based on system metrics.

- Improve cluster reliability and minimize downtime.

**3. Dataset Overview**

- Source: `kubernetes\_final\_dataset.csv`

- Features:

- CPU usage

- Memory usage

- Disk I/O

- Network latency

- Event logs

Target Variable: `event\_type` (categorical variable representing failure types).

**4. Project Workflow**

4.1 Data Preprocessing

- Handle missing values by replacing them with zero.

- Encode categorical variables using `LabelEncoder`.

4.2 Feature Selection

- Select key metrics such as CPU, memory, disk, and network usage.

4.3 Data Splitting & Normalization

- Split the dataset into 80% training and 20% testing.

- Scale numerical features using `StandardScaler`.

4.4 Model Training

- Train a Random Forest Classifier with 100 estimators.

4.5 Model Evaluation

- Compute accuracy, precision, recall, and F1-score.

**5. Implementation**

5.1 Load & Preprocess Data

# Load dataset

df = pd.read\_csv('kubernetes\_final\_dataset.csv')

# Handle missing values

df.fillna(0, inplace=True)

# Encode categorical variables

label\_encoders = {}

for column in ['event\_type', 'pod\_status', 'deployment\_strategy', 'scaling\_policy', 'traffic\_label']:

le = LabelEncoder()

df[column] = le.fit\_transform(df[column])

label\_encoders[column] = le

5.2 Train the Model

# Select features

features = ['cpu\_usage', 'memory\_usage', 'network\_latency', 'disk\_io']

X = df[features]

y = df['event\_type']

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Scale features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Train Random Forest model

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

5.3 Model Evaluation

# Predict on test data

y\_pred = model.predict(X\_test)

# Evaluate model

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

**6. Results**

- Accuracy: 32.4%

- Precision, Recall, and F1-score Analysis:

| Class | Precision | Recall | F1-score | Support |
| --- | --- | --- | --- | --- |
| 0 | 0.33 | 0.38 | 0.36 | 255 |
| 1 | 0.31 | 0.28 | 0.29 | 239 |
| 2 | 0.33 | 0.31 | 0.32 | 248 |

**7. Decision Tree Visualization**

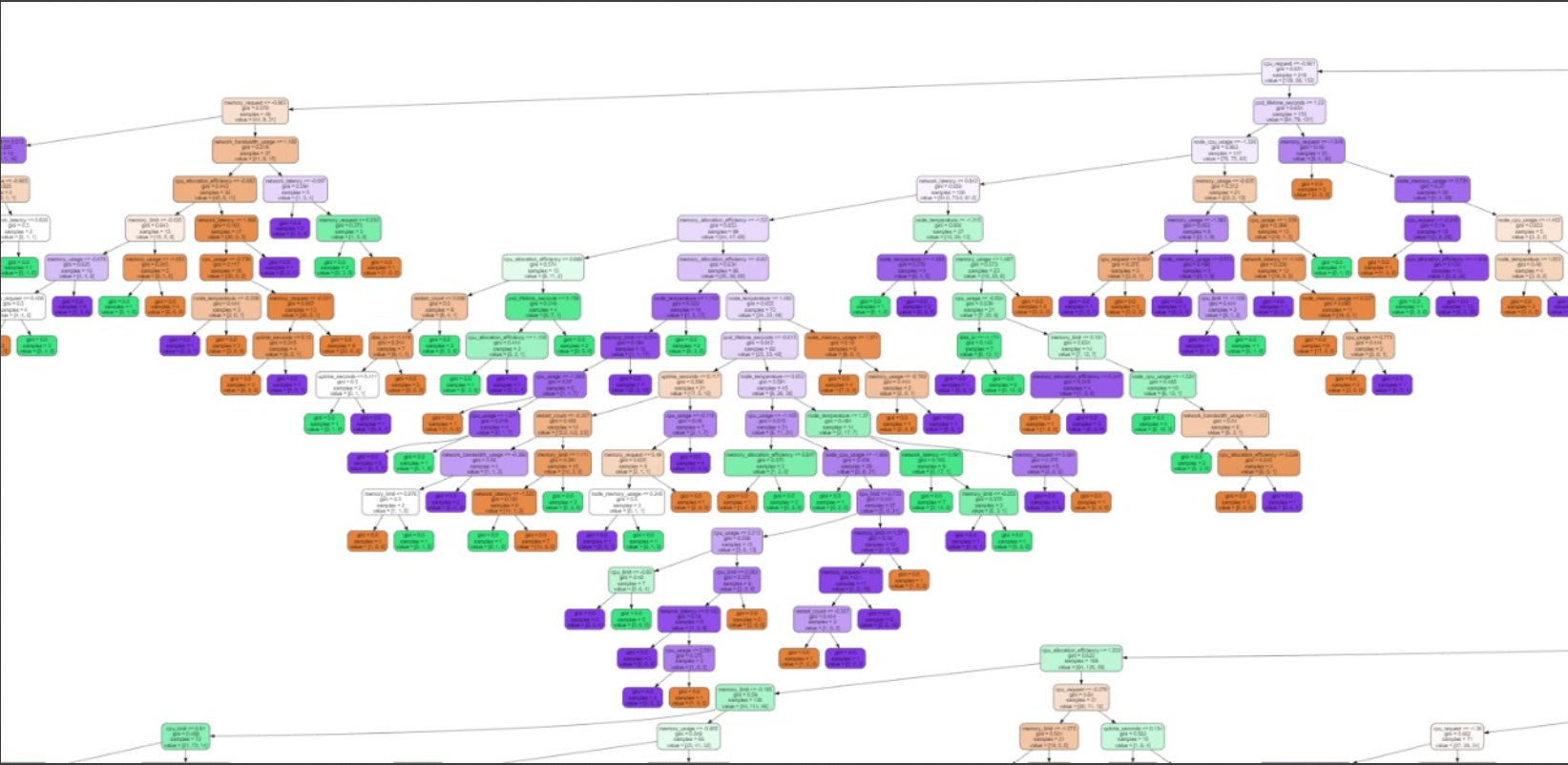
The following tree represents the first decision tree from the Random Forest model:

# Plot the first decision tree from the Random Forest

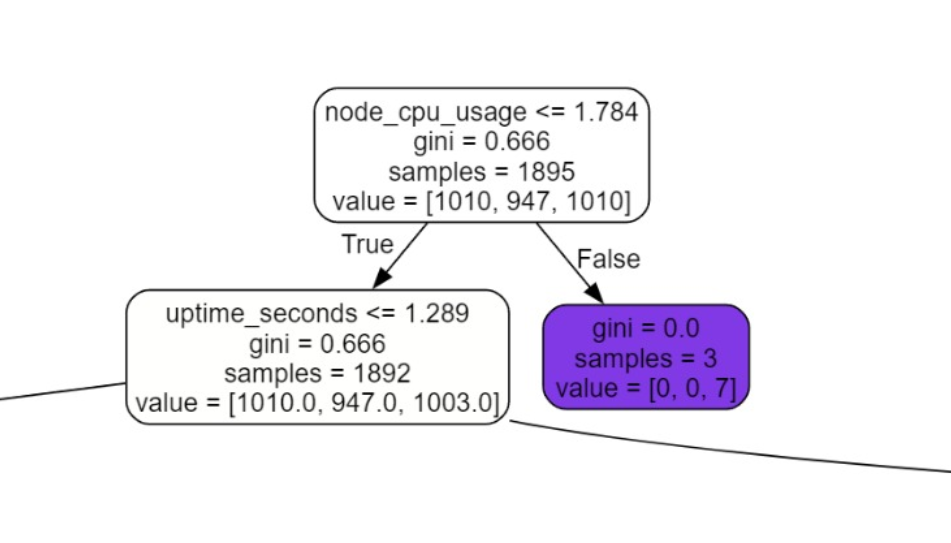
plt.figure(figsize=(20, 10))

tree.plot\_tree(model.estimators\_[0], feature\_names=features, filled=True, rounded=True)

plt.show()



Example-



**8. Deployment & Future Improvements**

- Save the trained model using `joblib` for future use.

- Improve accuracy by tuning hyperparameters.

- Implement real-time failure detection in Kubernetes clusters.

Code:(for future use )

# Load the saved model and scaler

model = joblib.load('kubernetes\_failure\_prediction\_model.pkl')

scaler = joblib.load('scaler.pkl')

# Use the model for predictions

new\_predictions = model.predict(X\_test)

**9. Conclusion**

- Successfully built a Kubernetes failure prediction model.

- Identified key resource usage metrics affecting failures.

- The model can be further improved for better accuracy.

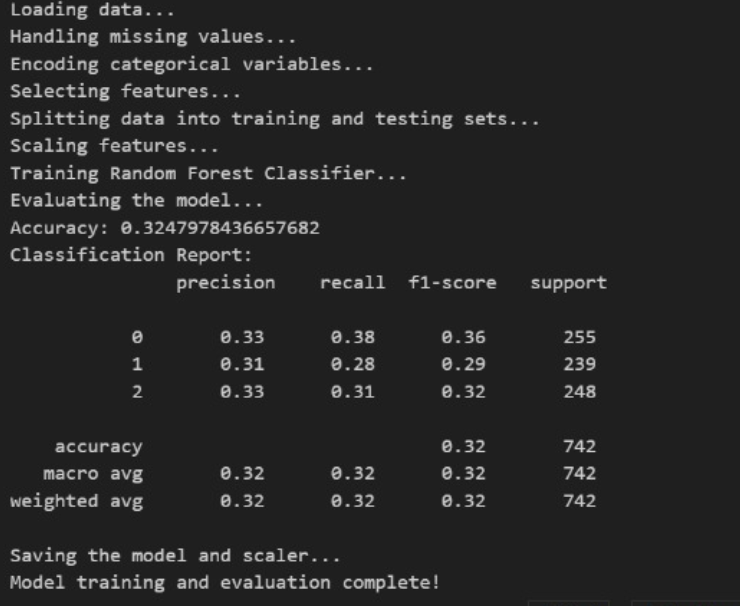
**10. References**

- Scikit-Learn Documentation : [https://scikit-learn.org/stable/](https://scikit-learn.org/stable/)

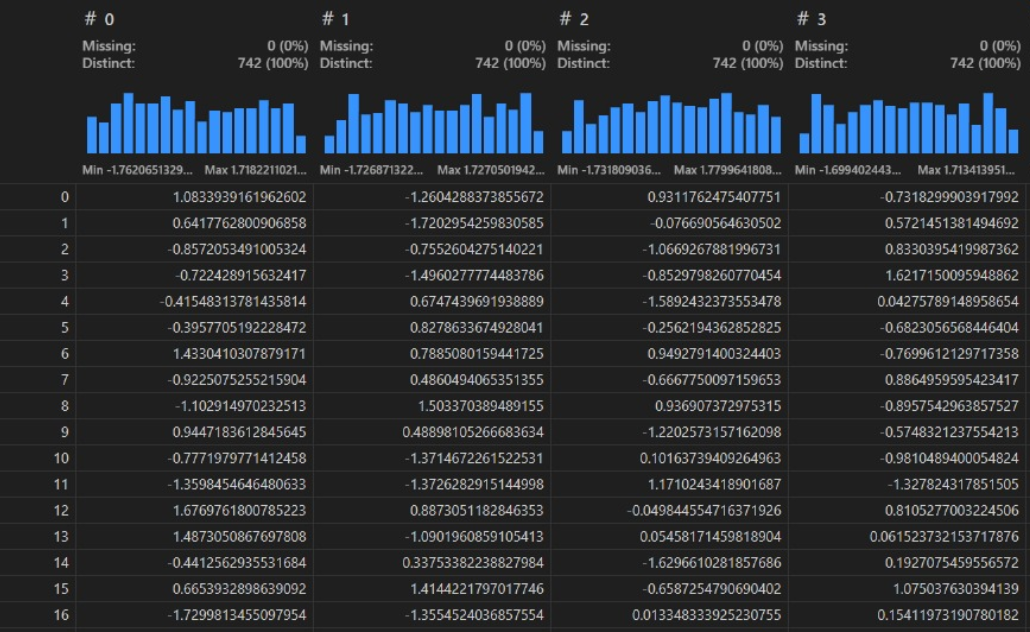
- Kubernetes Official Docs: [https://kubernetes.io/docs/](https://kubernetes.io/docs/)

Images

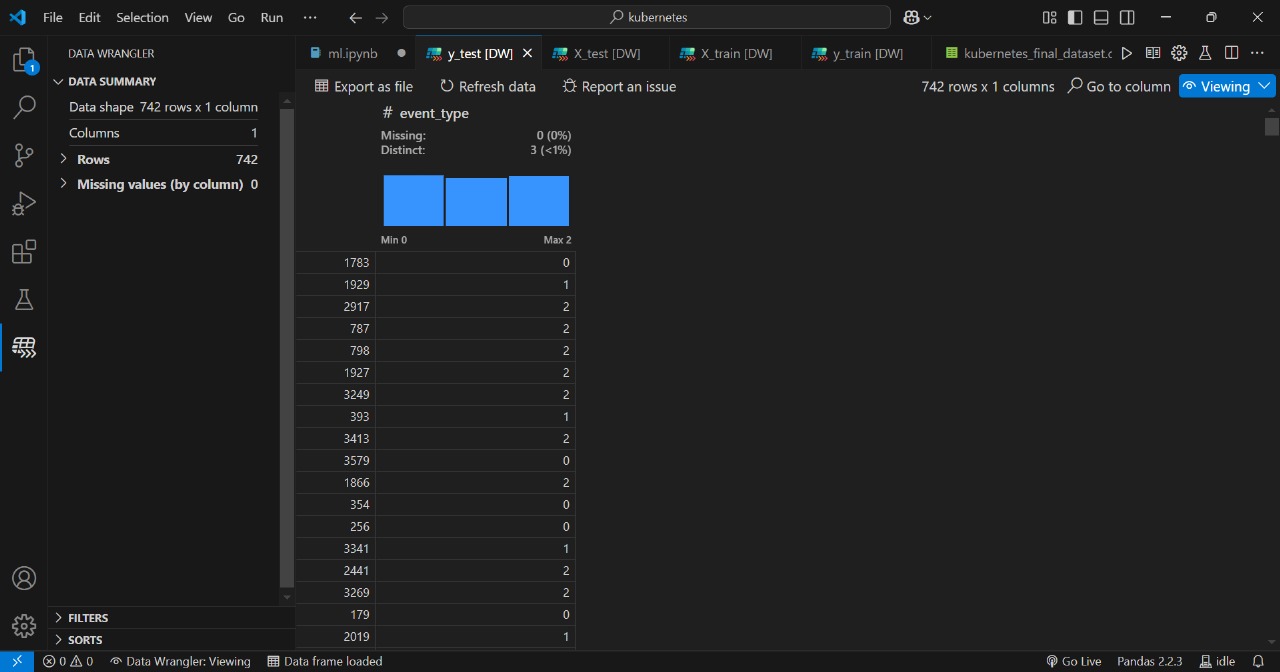
1. Data Summary - output



2. Model Evaluation



3. Data Wrangler



4. Explorer