

Use of AI

[1] Tool: Gemini

Prompt: Confusion matrix how to code

Output:

That's a great next step! The confusion matrix is the most fundamental way to break down your model's performance and understand *why* it gets certain metrics (like Precision and Recall).

To implement the confusion matrix, we need to use the `confusion_matrix` function from `scikit-learn` and then typically use a visualization library like `seaborn` to display it clearly as a heatmap.

```
from sklearn.metrics import confusion_matrix
# ... after training the model and getting predictions ...
cm = confusion_matrix(y_test, y_pred_knn_gs)

import seaborn as sns
# ... plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', ...)
plt.title('Confusion Matrix for KNN (GridSearch)')
# ... labels and savefig ...
```

[2] Tool: Gemini

Prompt: Confusion matrix how to code

Output:

Writing a clean, robust method to convert descriptive time strings into a standardized number (days) is a crucial step in preparing time-based categorical features.

I will provide the function, `age_to_days`, in a self-contained Python file along with an explanation of how it works and examples of how to apply it using pandas.

How the Method Works

The `age_to_days` method uses string manipulation and conditional logic to perform the conversion:

1. **Check for Nulls:** It first checks if the input is `NaN` or `None` and returns `NaN` if so, preventing errors later.
2. **Standardize:** It converts the string to lowercase ("`2 Years`" becomes "`2 years`") and removes surrounding spaces (`.strip()`).
3. **Split:** It uses `age_str.split(' ', 1)` to separate the number (`value`) from the time unit (`unit`).
4. **Convert:** It checks if the `unit` string contains "`day`", "`week`", "`month`", or "`year`", and applies the corresponding multiplication factor:
 - o `* 7` for weeks
 - o `* 30` for months (approximation)
 - o `* 365` for years (approximation)
5. **Apply to DataFrame:** You use the `.apply()` method on the DataFrame column to execute this function on every row, creating your new continuous numerical feature (`Age_in_Days`).

```
import pandas as pd

import numpy as np

def age_to_days(age_str):

    """
```

Converts a descriptive age string (e.g., '2 years', '5 months', '3 weeks')

into a continuous numerical value representing the total age in days.

This function handles nulls and uses standard approximations for conversion:

1 week = 7 days

1 month = 30 days

1 year = 365 days

Args:

age_str (str): The age string from the 'Age upon Outcome' column.

Returns:

float: The age converted to days, or np.nan if conversion fails or input is null.

```
"""
```

```
# 1. Handle Null or NaN input
```

```
if pd.isnull(age_str):
```

```
    return np.nan
```

```
# 2. Clean and standardize the string format
```

```
age_str = str(age_str).lower().strip()
```

```
try:

    # Expects format like "VALUE UNIT", e.g., "5 months"

    parts = age_str.split(' ', 1)

    if len(parts) != 2:

        return np.nan # Not the expected format (e.g., "unknown")

    value = float(parts[0])

    unit = parts[1]

except ValueError:

    return np.nan # Value is not a number

# 3. Perform the conversion based on the unit

if 'day' in unit:

    return value

elif 'week' in unit:

    return value * 7

elif 'month' in unit:

    return value * 30

elif 'year' in unit:

    return value * 365
```

```

    else:

        # Handles unexpected units, returning NaN

        return np.nan


# --- Example Usage ---

print("--- Example Conversion Test ---")

test_ages = [

    "5 months",

    "2 weeks",

    "10 years",

    "1 day",

    "NaN",

    "unknown value"

]


for age in test_ages:

    days = age_to_days(age)

    print(f"'{age}' converts to: {days:.0f} days (or {days} if using float)")


# --- Example with a Pandas Series (Simulating your DataFrame column) ---

print("\n--- Pandas Series Application ---")

```

```
data = {  
    'Age_String': ["4 months", "1 year", "3 weeks", np.nan, "2 days"]  
}  
  
df_example = pd.DataFrame(data)  
  
df_example['Age_in_Days'] = df_example['Age_String'].apply(age_to_days)  
  
print(df_example)
```