INTERMEDIATE TASKS

TASK NO 1 Build a model with cross - validation

```
import pandas as pd
from sklearn.model_selection import cross_val_score, KFold
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression

data = load_iris()
X, y = data.data, data.target

model = LogisticRegression(max_iter=200)
kfold = KFold(n_splits=5, shuffle=True, random_state=42)

cv_scores = cross_val_score(model, X, y, cv=kfold, scoring='accuracy')

print("📊 Cross-validation scores (per fold):", cv_scores)
print(f"✅ Mean Accuracy: {cv_scores.mean():.4f}")
print(f"📈 Standard Deviation: {cv_scores.std():.4f}")
```

```
📊 Cross-validation scores (per fold): [1.         1.         0.93333333 0.96666667 0.96666667]
✅ Mean Accuracy: 0.9733
📈 Standard Deviation: 0.0249
```

TASK NO 2 Preprocesss data for Machine learning

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.datasets import load_breast_cancer

data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)

X['category'] = np.where(X['mean radius'] > X['mean radius'].mean(), 'High', 'Low')

numeric_features = X.select_dtypes(include=[np.number]).columns.tolist()
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

categorical_features = ['category']
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ])

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

print("✅ Dataset loaded and preprocessing pipeline is ready")
print("Training shape:", X_train.shape)
print("Testing shape:", X_test.shape)
```

```
✅ Dataset loaded and preprocessing pipeline is ready
Training shape: (455, 31)
Testing shape: (114, 31)
```

TASK NO 3 Create a classification Report

```
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from rich.console import Console
from rich.table import Table
from rich import box

clf = Pipeline(steps=[('preprocessor', preprocessor),
                      ('classifier', RandomForestClassifier(random_state=42))])
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

report = classification_report(y_test, y_pred, output_dict=True)

console = Console()

table = Table(
    title="🌈 Classification Report",
    title_style="bold white on blue",
```

```
    header_style="bold white on black",
    style="bold",
    box=box.DOUBLE_EDGE
)

table.add_column("Class", justify="center", style="bold cyan")
table.add_column("Precision", justify="center", style="bold green")
table.add_column("Recall", justify="center", style="bold yellow")
table.add_column("F1-Score", justify="center", style="bold magenta")
table.add_column("Support", justify="center", style="bold red")

row_colors = ["on grey15", "on grey7"]
i = 0

for label, metrics in report.items():
    if isinstance(metrics, dict):
        table.add_row(
            f"[cyan]{label}[/cyan]",
            f"[green]{metrics['precision']:.2f}[/green]",
            f"[yellow]{metrics['recall']:.2f}[/yellow]",
            f"[magenta]{metrics['f1-score']:.2f}[/magenta]",
            f"[red]{metrics['support']}[/red]",
            style=row_colors[i % 2]
        )
        i += 1

console.print(table)
```

🌈 Classification Report

Class	Precision	Recall	F1-Score	Support
0	0.98	0.93	0.95	43.0
1	0.96	0.99	0.97	71.0
macro avg	0.97	0.96	0.96	114.0
weighted avg	0.97	0.96	0.96	114.0

Start coding or generate with AI.