Model Development Phase Template

Date	25 March 2025
Team ID	SWTID1749641473
Project Title	Early Prediction for Chronic Kidney Disease Detection: A Progressive Approach to Health Management
Maximum Marks	5 Marks

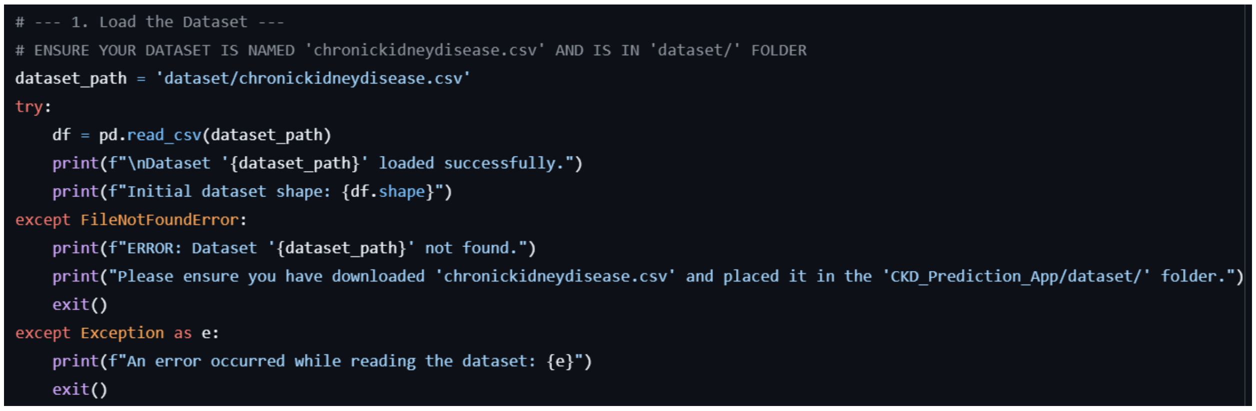
Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include classification reports, accuracy, and confusion matrices for multiple models, presented through respective screenshots.

Initial Model Training Code:

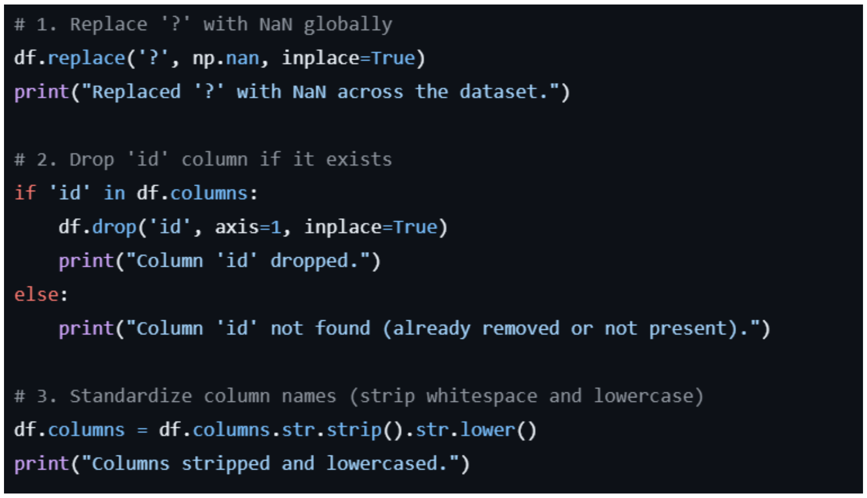
The machine learning pipeline was implemented in ml_pipeline.py. The key steps included:

- Data loading:

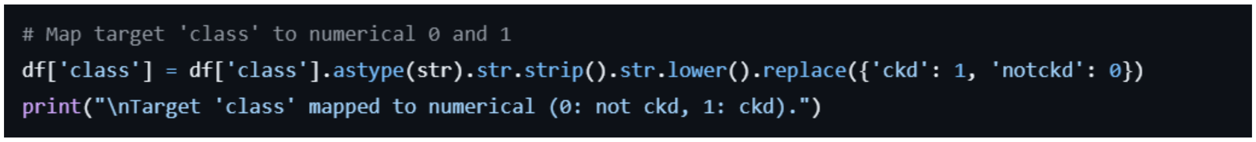


```python
# --- 1. Load the Dataset ---
# ENSURE YOUR DATASET IS NAMED 'chronickidneydisease.csv' AND IS IN 'dataset/' FOLDER
dataset_path = 'dataset/chronickidneydisease.csv'
try:
    df = pd.read_csv(dataset_path)
    print(f"\nDataset '{dataset_path}' loaded successfully.")
    print(f"Initial dataset shape: {df.shape}")
except FileNotFoundError:
    print(f"ERROR: Dataset '{dataset_path}' not found.")
    print("Please ensure you have downloaded 'chronickidneydisease.csv' and placed it in the 'CKD_Prediction_App/dataset/' folder.")
    exit()
except Exception as e:
    print(f"An error occurred while reading the dataset: {e}")
    exit()
```

- Initial Cleaning:

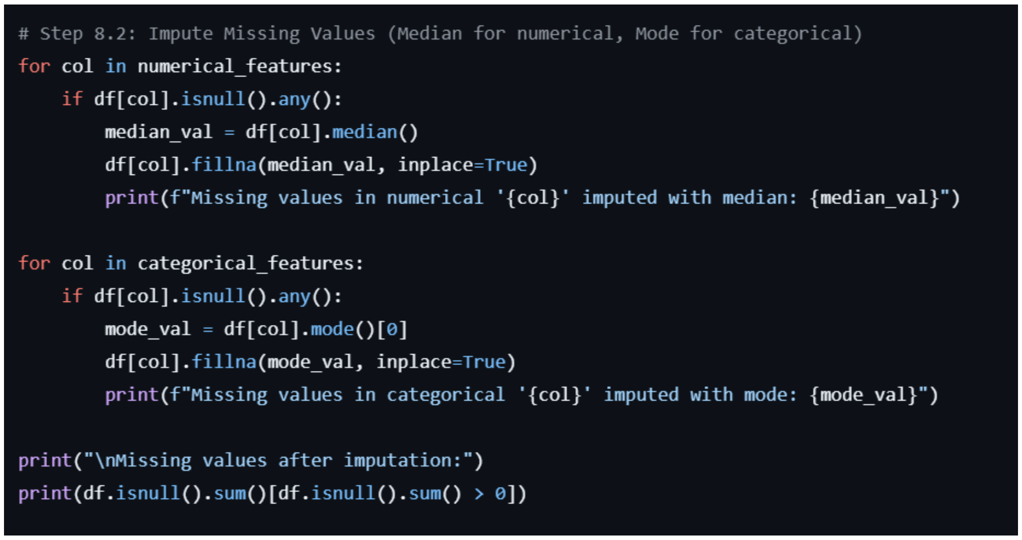


```
# 1. Replace '?' with NaN globally
df.replace('?', np.nan, inplace=True)
print("Replaced '?' with NaN across the dataset.")

# 2. Drop 'id' column if it exists
if 'id' in df.columns:
    df.drop('id', axis=1, inplace=True)
    print("Column 'id' dropped.")
else:
    print("Column 'id' not found (already removed or not present).")

# 3. Standardize column names (strip whitespace and lowercase)
df.columns = df.columns.str.strip().str.lower()
print("Columns stripped and lowercased.")
```



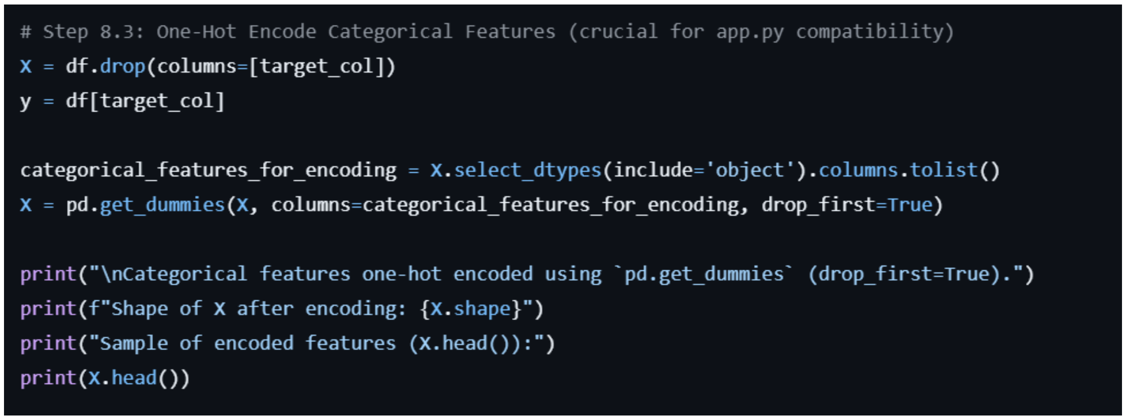
```
# Map target 'class' to numerical 0 and 1
df['class'] = df['class'].astype(str).str.strip().str.lower().replace({'ckd': 1, 'notckd': 0})
print("\nTarget 'class' mapped to numerical (0: not ckd, 1: ckd).")
```

- Handling Missing Values:



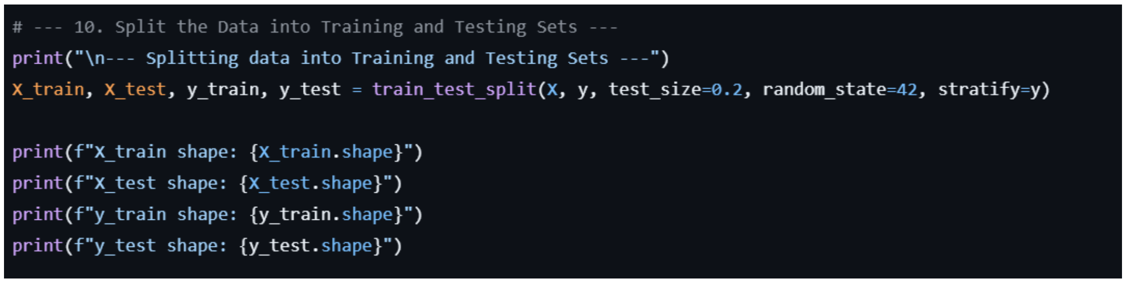
```
# Step 8.2: Impute Missing Values (Median for numerical, Mode for categorical)
for col in numerical_features:
    if df[col].isnull().any():
        median_val = df[col].median()
        df[col].fillna(median_val, inplace=True)
        print(f"Missing values in numerical '{col}' imputed with median: {median_val}")

for col in categorical_features:
    if df[col].isnull().any():
        mode_val = df[col].mode()[0]
        df[col].fillna(mode_val, inplace=True)
        print(f"Missing values in categorical '{col}' imputed with mode: {mode_val}")

print("\nMissing values after imputation:")
print(df.isnull().sum()[df.isnull().sum() > 0])
```

- <u>Feature Engineering:</u>



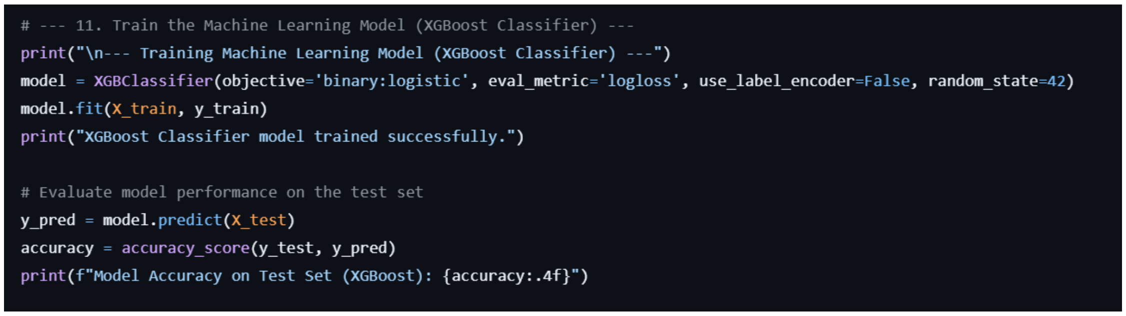
```python
# Step 8.3: One-Hot Encode Categorical Features (crucial for app.py compatibility)
X = df.drop(columns=[target_col])
y = df[target_col]

categorical_features_for_encoding = X.select_dtypes(include='object').columns.tolist()
X = pd.get_dummies(X, columns=categorical_features_for_encoding, drop_first=True)

print("\nCategorical features one-hot encoded using `pd.get_dummies` (drop_first=True).")
print(f"Shape of X after encoding: {X.shape}")
print("Sample of encoded features (X.head()):")
print(X.head())
```

- <u>Train-Test Split:</u>



```python
# --- 10. Split the Data into Training and Testing Sets ---
print("\n--- Splitting 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, stratify=y)

print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

- <u>Model Training:</u>



```python
# --- 11. Train the Machine Learning Model (XGBoost Classifier) ---
print("\n--- Training Machine Learning Model (XGBoost Classifier) ---")
model = XGBClassifier(objective='binary:logistic', eval_metric='logloss', use_label_encoder=False, random_state=42)
model.fit(X_train, y_train)
print("XGBoost Classifier model trained successfully.")

# Evaluate model performance on the test set
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy on Test Set (XGBoost): {accuracy:.4f}")
```

Model Training and Evaluation Report:

Model	**Accuracy**	**F1-Score**	**Confusion Matrix**
XGBoost	98.75%	99%	[[30 0] [1 49]] Actual Not CKD: 30 TP, 0 FNActual CKD: 49 TP, 1 FN

<u>Classification Report:</u>

Class	**Precision**	**Recall**	**F1-score**	**Support**
Not CKD	0.97	1.00	0.98	30
CKD	1.00	0.98	0.99	50
Macro avg	0.98	0.99	0.99	80
Weighted avg	0.99	0.99	0.99	80