Disease Prediction from Medical Data

A Comprehensive Explanation of the AI Model Notebook

This document presents a detailed explanation of the Jupyter Notebook implementation for predicting diseases using machine learning models. Each section of code is broken down into clear parts, with professional descriptions of its purpose, functionality, and importance in the workflow.

Cell 1: Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Explanation:

This cell imports the fundamental Python libraries required for data analysis, visualization, and handling warnings. NumPy and Pandas are used for numerical computations and data handling respectively. Matplotlib is utilized for creating plots, while rcParams and rainbow enhance visualization aesthetics. The inline command ensures plots are displayed within the notebook.

Cell 2: Code

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Explanation:

This cell imports utilities from Scikit-learn for data preprocessing and splitting. 'train_test_split' is used to divide the dataset into training and testing subsets, ensuring unbiased model evaluation. 'StandardScaler' normalizes the data, which improves performance for algorithms sensitive to scale.

Cell 3: Code

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

Explanation:

Here, multiple machine learning classifiers are imported from Scikit-learn, including K-Nearest Neighbors, Support Vector Classifier, Decision Tree, and Random Forest. These classifiers represent diverse algorithmic approaches for supervised learning, making the prediction system more robust.

Cell 4: Code

```
dataset = pd.read_csv('/content/Dataset for heart diseases.csv') # Make sure this is the correct
```

Explanation:

The dataset containing medical information is loaded using Pandas. The CSV file path is specified, and the dataset is stored in a DataFrame for further analysis and preprocessing.

Cell 5: Code

```
dataset.info()
```

Explanation:

The dataset.info() function is executed to display a concise summary of the dataset, including the number of entries, column names, data types, and non-null counts. This helps in understanding the structure and completeness of the data.

Cell 6: Code

```
dataset.describe()
```

Explanation:

The dataset.describe() function provides statistical insights such as mean, standard deviation, minimum, and maximum values for numerical features. This step assists in identifying potential outliers and understanding the general distribution of the data.

Cell 7: Code

```
rcParams['figure.figsize'] = 20, 14
plt.matshow(dataset.corr())
plt.yticks(np.arange(dataset.shape[1]), dataset.columns)
plt.xticks(np.arange(dataset.shape[1]), dataset.columns)
plt.colorbar()
```

Explanation:

A correlation matrix is generated using Matplotlib to examine relationships between different features. Correlation values are visualized using a heatmap, enabling identification of strongly correlated variables that may impact disease prediction.

Cell 8: Code

```
dataset.hist()
```

Explanation:

A histogram for each numerical column is plotted to visualize the distribution of individual features. Histograms help in understanding the spread, skewness, and presence of unusual patterns in the dataset.

Cell 9: Code

```
rcParams['figure.figsize'] = 8,6
plt.bar(dataset['condition'].unique(), dataset['condition'].value_counts(), color = ['red', 'gre
plt.xticks([0, 1])
plt.xlabel('Target Classes')
plt.ylabel('Count')
plt.title('Count of each Target Class')
```

Explanation:

This visualization plots the distribution of the target variable (disease presence). A bar chart displays the counts of positive and negative cases, providing a clear understanding of class balance within the dataset.

Cell 10: Code

```
dataset = pd.get_dummies(dataset, columns = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'c
```

Explanation:

Categorical columns are transformed into numerical format using one-hot encoding (via Pandas get_dummies). This step is crucial because machine learning models require numerical input. Encoding ensures categorical attributes are represented in a binary form without introducing ordinal bias.

Cell 11: Code

```
standardScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset[columns_to_scale] = standardScaler.fit_transform(dataset[columns_to_scale])
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 12: Code

```
y = dataset['condition']
X = dataset.drop(['condition'], axis = 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 0)
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 13: Code

```
knn_scores = []
for k in range(1,21):
    knn_classifier = KNeighborsClassifier(n_neighbors = k)
    knn_classifier.fit(X_train, y_train)
    knn_scores.append(knn_classifier.score(X_test, y_test))
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 14: Code

```
plt.plot([k for k in range(1, 21)], knn_scores, color = 'red')
for i in range(1,21):
    plt.text(i, knn_scores[i-1], (i, knn_scores[i-1]))
plt.xticks([i for i in range(1, 21)])
```

```
plt.xlabel('Number of Neighbors (K)')
plt.ylabel('Scores')
plt.title('K Neighbors Classifier scores for different K values')
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 15: Code

```
print("The score for K Neighbors Classifier is {}% with {} nieghbors.".format(knn_scores[7]*100,
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 16: Code

```
svc_scores = []
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
for i in range(len(kernels)):
    svc_classifier = SVC(kernel = kernels[i])
    svc_classifier.fit(X_train, y_train)
    svc_scores.append(svc_classifier.score(X_test, y_test))
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 17: Code

```
colors = rainbow(np.linspace(0, 1, len(kernels)))
plt.bar(kernels, svc_scores, color = colors)
for i in range(len(kernels)):
    plt.text(i, svc_scores[i], svc_scores[i])
plt.xlabel('Kernels')
plt.ylabel('Scores')
plt.title('Support Vector Classifier scores for different kernels')
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 18: Code

```
print("The score for Support Vector Classifier is {}% with {} kernel.".format(svc_scores[0]*100,
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 19: Code

```
dt_scores = []
```

```
for i in range(1, len(X.columns) + 1):
    dt_classifier = DecisionTreeClassifier(max_features = i, random_state = 0)
    dt_classifier.fit(X_train, y_train)
    dt_scores.append(dt_classifier.score(X_test, y_test))
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 20: Code

```
plt.plot([i for i in range(1, len(X.columns) + 1)], dt_scores, color = 'green')
for i in range(1, len(X.columns) + 1):
    plt.text(i, dt_scores[i-1], (i, dt_scores[i-1]))
plt.xticks([i for i in range(1, len(X.columns) + 1)])
plt.xlabel('Max features')
plt.ylabel('Scores')
plt.title('Decision Tree Classifier scores for different number of maximum features')
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 21: Code

```
print("The score for Decision Tree Classifier is {}% with {} maximum features.".format(dt_scores
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 22: Code

```
rf_scores = []
estimators = [10, 100, 200, 500, 1000]
for i in estimators:
    rf_classifier = RandomForestClassifier(n_estimators = i, random_state = 0)
    rf_classifier.fit(X_train, y_train)
    rf_scores.append(rf_classifier.score(X_test, y_test))
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 23: Code

```
colors = rainbow(np.linspace(0, 1, len(estimators)))
plt.bar([i for i in range(len(estimators))], rf_scores, color = colors, width = 0.8)
for i in range(len(estimators)):
    plt.text(i, rf_scores[i], rf_scores[i])
plt.xticks(ticks = [i for i in range(len(estimators))], labels = [str(estimator) for estimator i
plt.xlabel('Number of estimators')
plt.ylabel('Scores')
plt.title('Random Forest Classifier scores for different number of estimators')
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 24: Code

```
print("The score for Random Forest Classifier is {}% with {} estimators.".format(rf_scores[1]*10
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 25: Code

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 27: Code

```
user_input = {}
for col in X.columns:
    while True:
        try:
            value = input(f"Enter value for {col}: ")
            # Attempt to convert to appropriate type based on column data type in X_train.
            # This is a basic attempt, more robust type handling might be needed.
            if X_train[col].dtype == 'float64':
                user_input[col] = float(value)
            elif X_train[col].dtype == 'int64':
                 user_input[col] = int(value)
            else: # Assuming boolean columns from get_dummies
                user_input[col] = bool(int(value)) # Convert to int first then bool

            break # Exit the inner loop if input is valid
        except ValueError:
            print("Invalid input. Please enter a number.")
        except Exception as e:
            print(f"An unexpected error occurred: {e}")
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 29: Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
```

```
import warnings
warnings.filterwarnings('ignore')
```
Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 31: Code

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```
Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 33: Code

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```
Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 35: Code

```
dataset = pd.read_csv('/content/heart_disease_and_some_scikit_learn_magic.csv') # Make sure this
```
Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 37: Code

```
dataset.info()
```
Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 38: Code

```
knn_prediction = knn_classifier.predict(user_df)
svc_prediction = svc_classifier.predict(user_df)
dt_prediction = dt_classifier.predict(user_df)
rf_prediction = rf_classifier.predict(user_df)
```
Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 39: Code

```
print(f"KNN Prediction: {knn_prediction[0]}")
print(f"SVC Prediction: {svc_prediction[0]}")
print(f"Decision Tree Prediction: {dt_prediction[0]}")
print(f"Random Forest Prediction: {rf_prediction[0]}")
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 40: Code

```
user_df = pd.DataFrame([user_input])
user_df = user_df[X.columns]

columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
user_df[columns_to_scale] = standardScaler.transform(user_df[columns_to_scale])

display(user_df)
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 41: Code

```
user_input = {}
for col in X.columns:
    while True:
        try:
            value = input(f"Enter value for {col}: ")
            # Attempt to convert to appropriate type based on column data type in X_train.
            # This is a basic attempt, more robust type handling might be needed.
            if X_train[col].dtype == 'float64':
                user_input[col] = float(value)
            elif X_train[col].dtype == 'int64':
                 user_input[col] = int(value)
            else: # Assuming boolean columns from get_dummies
                user_input[col] = bool(int(value)) # Convert to int first then bool

            break # Exit the inner loop if input is valid
        except ValueError:
            print("Invalid input. Please enter a number.")
        except Exception as e:
            print(f"An unexpected error occurred: {e}")
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 42: Code

```
user_df = pd.DataFrame([user_input])
user_df = user_df[X.columns]

columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
user_df[columns_to_scale] = standardScaler.transform(user_df[columns_to_scale])

display(user_df)
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 43: Code

```
knn_prediction = knn_classifier.predict(user_df)
svc_prediction = svc_classifier.predict(user_df)
dt_prediction = dt_classifier.predict(user_df)
rf_prediction = rf_classifier.predict(user_df)
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.

Cell 44: Code

```
print(f"KNN Prediction: {knn_prediction[0]}")
print(f"SVC Prediction: {svc_prediction[0]}")
print(f"Decision Tree Prediction: {dt_prediction[0]}")
print(f"Random Forest Prediction: {rf_prediction[0]}")
```

Explanation:

This cell performs an important step in the pipeline. It contributes towards preprocessing, model training, evaluation, or visualization depending on its context.