**Revolutionizing Early Diagnosis: K-NN Classifier for Parkinson's Disease Detection**

Article Reference:  
Lahmiri, S., Shmuel, A. (2023). "Accurate detection of Parkinson's disease using ensemble learning and voice signal features." Biomedical Signal Processing and Control, 80, 104363.

Summary:

Application:  
This study uses a K-NN classifier in an ensemble learning strategy to detect Parkinson's disease (PD) from voice signal attributes. Early detection of Parkinson's disease is critical for effective therapy, and this non-invasive approach presents a possible answer.

Data:  
The study used the Parkinson's Disease Voice Dataset from the UCI Machine Learning Repository, which included 195 recordings (31 from PD patients and 164 from healthy individuals). Each recording is identified by 23 voice measurement features.

Training Protocol:

1. Use the ReliefF method to choose the most relevant voice measurements.
2. Normalize data to guarantee consistent scaling across all characteristics.
3. Added a K-NN classifier to an ensemble learning framework, alongside Random Forest and Support Vector Machine.
4. Use 10-fold cross-validation to evaluate model performance and generalizability.

Results:  
The ensemble model, which included the K-NN classifier, produced impressive results:

* Accuracy: 99.49%
* Sensitivity: 100%
* Specificity: 99.39%
* F1-score: 0.9947

These findings outperform earlier research, indicating the effectiveness of combining K-NN with other classifiers in an ensemble approach to PD diagnosis.

Python Script:

Here's a functioning Python script that constructs a K-NN classifier for Parkinson's disease identification with the provided dataset:

| # Separate features and target  X = data.drop(['name', 'status'], axis=1)  y = data['status']  # Feature selectionimport numpy as np  import pandas as pd  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import StandardScaler  from sklearn.neighbors import KNeighborsClassifier  from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report  from sklearn.feature\_selection import SelectKBest, f\_classif  # Load the dataset  url = "https://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/parkinsons.data"  data = pd.read\_csv(url)  selector = SelectKBest(f\_classif, k=10)  X\_selected = selector.fit\_transform(X, y)  # Split the data  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_selected, y, test\_size=0.2, random\_state=42)  # Normalize the features  scaler = StandardScaler()  X\_train\_scaled = scaler.fit\_transform(X\_train)  X\_test\_scaled = scaler.transform(X\_test)  # Train K-NN classifier  knn = KNeighborsClassifier(n\_neighbors=5)  knn.fit(X\_train\_scaled, y\_train)  # Make predictions  y\_pred = knn.predict(X\_test\_scaled)  # Evaluate the model  accuracy = accuracy\_score(y\_test, y\_pred)  conf\_matrix = confusion\_matrix(y\_test, y\_pred)  class\_report = classification\_report(y\_test, y\_pred)  print(f"Accuracy: {accuracy:.4f}")  print("\nConfusion Matrix:")  print(conf\_matrix)  print("\nClassification Report:")  print(class\_report) |
| --- |

This script shows the major steps in developing a K-NN classifier for Parkinson's disease detection:

1. c
2. Here's a modification to our Python script that uses some of these sophisticated techniques:

| from sklearn.ensemble import RandomForestClassifier  from sklearn.svm import SVC  from sklearn.model\_selection import cross\_val\_score  from sklearn.ensemble import VotingClassifier  from sklearn.model\_selection import GridSearchCV  # ... (previous code for data loading and preprocessing)  # Hyperparameter tuning for K-NN  param\_grid = {'n\_neighbors': [3, 5, 7, 9, 11],  'weights': ['uniform', 'distance'],  'metric': ['euclidean', 'manhattan']}  grid\_search = GridSearchCV(KNeighborsClassifier(), param\_grid, cv=5)  grid\_search.fit(X\_train\_scaled, y\_train)  best\_knn = grid\_search.best\_estimator\_  # Create ensemble  rf = RandomForestClassifier(n\_estimators=100, random\_state=42)  svm = SVC(probability=True, random\_state=42)  ensemble = VotingClassifier(  estimators=[('knn', best\_knn), ('rf', rf), ('svm', svm)],  voting='soft'  )  # Train ensemble  ensemble.fit(X\_train\_scaled, y\_train)  # Make predictions  y\_pred = ensemble.predict(X\_test\_scaled)  # Evaluate using cross-validation  cv\_scores = cross\_val\_score(ensemble, X\_selected, y, cv=10)  print(f"Cross-validation accuracy: {cv\_scores.mean():.4f} (+/- {cv\_scores.std() \* 2:.4f})") |
| --- |

# ... (previous code for final evaluation and printing results)

This extended script showcases:

1. Tuning hyperparameters in the K-NN classifier.
2. Ensemble learning using K-NN, Random Forest, and SVM.
3. Use 10-fold cross-validation to ensure accurate performance estimation.

Future Work:

1. Apply the ReliefF algorithm for feature selection.
2. Add SHAP value analysis to improve model interpretability.
3. Integrate additional voice features or biomarkers to improve PD identification.
4. Examine the effectiveness of deep learning models, such as CNNs or RNNs, on raw voice data.
5. Conduct prospective research to assess the model's effectiveness in clinical settings.

This complete approach not only replicates the essence of the original work, but also provides a solid foundation for future research and progress in early Parkinson's disease diagnosis utilizing machine learning techniques.