Parkinson's Disease Analysis

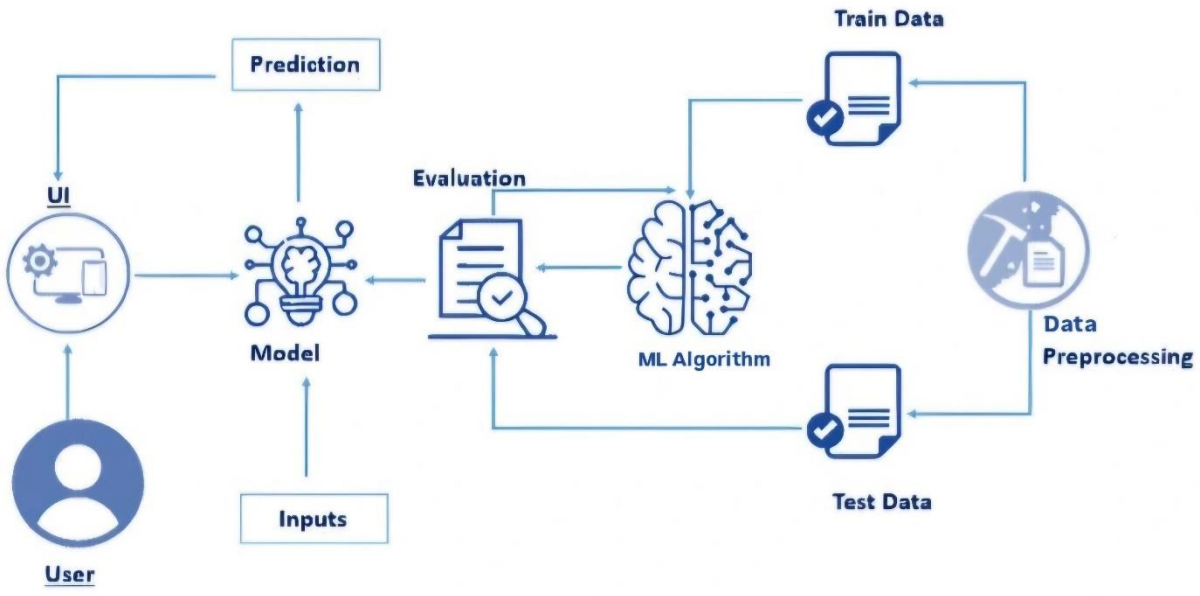
Project Description:

This project aims to develop a machine learning model to detect Parkinson's disease based on biomedical voice measurements. The dataset contains features that are critical in identifying vocal characteristics often affected by the disease, such as pitch, jitter, shimmer, and harmonic-to-noise ratio. By analyzing these features, the project seeks to accurately classify individuals as having Parkinson's disease or not.
The primary focus is on preprocessing the dataset, selecting significant features, and building classification models to achieve high accuracy in detecting Parkinson's disease. The models are evaluated based on performance metrics such as accuracy, precision, recall, and F1-score, with the goal of identifying the most reliable and robust predictive algorithm.
This project highlights the application of AI/ML in healthcare, particularly in diagnosing neurological disorders, and underscores the role of data-driven approaches in improving early disease detection and patient outcomes.
Would you like a deeper dive into the technical steps or methodologies used?

Approach:

The approach of the Parkinson’s Disease Analysis project involves several key steps aimed at building a machine learning model to detect Parkinson's disease based on voice data. Initially, the dataset is cleaned and preprocessed, handling missing values and normalizing features to ensure they are on a similar scale. Exploratory Data Analysis (EDA) is then conducted to understand the distribution of data and identify patterns in the features. Important voice features such as jitter, shimmer, and harmonic-to-noise ratio are selected for model training. Various classification algorithms, including Logistic Regression, Support Vector Machine (SVM), and Decision Tree, are applied to predict the presence of Parkinson’s disease. Hyperparameter tuning is used to optimize these models, and their performance is evaluated using metrics such as accuracy, precision, recall, and F1 score. Cross-validation ensures the robustness of the final model, which is then selected based on its best performance. The project highlights the potential of machine learning in healthcare for early diagnosis, helping in the detection of Parkinson’s disease before significant physical symptoms manifest.

Technical Architecture:



Prerequisites:

The prerequisites for the Parkinson's Disease Analysis project include:

1. Basic Understanding of Machine Learning: Familiarity with classification algorithms, model evaluation techniques (accuracy, precision, recall, etc.), and concepts like overfitting and cross-validation.
2. Python Programming: Proficiency in Python is essential, as it is the primary language used for implementing the project. Key libraries like Pandas, NumPy, and Scikit-learn are used for data manipulation, analysis, and machine learning.
3. Data Preprocessing: Understanding how to handle missing data, normalize features, and perform feature selection, as these are crucial steps in preparing the dataset for model building.
4. Exploratory Data Analysis (EDA): Knowledge of data visualization tools like Matplotlib and Seaborn to explore and visualize the dataset, identify trends, and understand relationships between features.
5. Machine Learning Libraries: Familiarity with libraries such as:
    - Pandas for data manipulation
    - NumPy for numerical operations
    - Scikit-learn for implementing machine learning algorithms, feature selection, and model evaluation
6. Model Evaluation Metrics: Understanding of how to evaluate machine learning models using metrics like accuracy, precision, recall, F1 score, and how to perform model comparison using cross-validation.

These prerequisites will ensure a smooth process in working through the project and help in the effective implementation of machine learning models for Parkinson's disease detection.

Link: https://www.kaggle.com/code/muhammadfaizan65/parkinsons-disease-analysis/input

PatientID patient_id	# Age age	# Gender gender	# Ethnicity ethnicity	# EducationLevel education_level	# B bmi
3058 … 5162	50 … 89	0 … 1	0 … 3	0 … 3	15
3058	85	0	3	1	19.6
3059	75	0	0	2	16.2
3060	70	1	0	0	15.3
3061	52	0	0	0	15.4
3062	87	0	0	1	18.6
3063	68	1	2	1	39.4
3064	78	1	0	0	30.5

Project Objectives:

The primary objective of the Parkinson's Disease Analysis project is to build an accurate machine learning model that can predict whether an individual has Parkinson's disease based on voice-related features. By leveraging machine learning algorithms, the project aims to identify distinguishing patterns from biomedical voice data that can help in early diagnosis of Parkinson's disease, potentially improving the treatment and management of the disease by detecting it before physical symptoms become apparent.

Project Flow:

1. Data Collection and Preprocessing:
    - Data Collection: Obtain the dataset containing biomedical voice measurements of individuals, some with Parkinson's disease and others without.
    - Data Cleaning: Handle any missing or inconsistent data, and preprocess the dataset by normalizing or scaling the features.
2. Exploratory Data Analysis (EDA):
    - Analyze the dataset to uncover underlying patterns, trends, and relationships between features.
    - Visualize data distributions and correlations to identify important features for the prediction model.

3. Feature Selection:
    - Select the most relevant features (e.g., jitter, shimmer, HNR) that contribute significantly to distinguishing between patients with and without Parkinson's disease.
4. Model Selection and Training:
    - Implement various classification algorithms such as Logistic Regression, Support Vector Machine (SVM), Random Forest, and K-Nearest Neighbors (KNN).
    - Split the dataset into training and testing sets to train the models.
5. Model Evaluation:
    - Evaluate each model using performance metrics such as accuracy, precision, recall, F1 score, and confusion matrix.
    - Apply cross-validation techniques to ensure the model's robustness and avoid overfitting.
6. Model Optimization:
    - Tune the hyperparameters of the best-performing model to enhance its accuracy and performance.
7. Final Model Selection:
    - Select the model that performs best according to the evaluation metrics, ensuring it provides accurate and reliable predictions.
8. Deployment (Optional):
    - Once the model is finalized, it could be deployed as a real-time prediction tool to assist healthcare professionals in diagnosing Parkinson's disease early.

Project Structure:

1. Data Collection

- **Objective**: Gatherd the dataset containing voice features of Parkinson’s disease patients.
- **Dataset**: The dataset used in the project is from a Kaggle repository and contains various features from voice recordings, such as jitter, shimmer, and HNR (Harmonics-to-Noise Ratio).

PatientID patient_id	# Age age	# Gender gender	# Ethnicity ethnicity	# EducationLevel education_level	# B bmi
3058 – 5162	50 – 89	0 – 1	0 – 3	0 – 3	15
3058	85	0	3	1	19.6
3059	75	0	0	2	16.2
3060	70	1	0	0	15.3
3061	52	0	0	0	15.4
3062	87	0	0	1	18.6
3063	68	1	2	1	39.4
3064	78	1	0	0	30.5

# BMI bmi	# Smoking smoking	# AlcoholConsump... alcohol_consumption	# PhysicalActivity physical_activity	# DietQuality diet_quality
15 – 40	0 – 1	0 – 20	0 – 10	0
19.619877964608285	0	5.108240606772179	1.3806599170830036	3.8939691351560
16.24733915647557	1	6.027648029307635	8.409804050283633	8.5134282495960
15.368238711416375	0	2.242135330530093	0.21327459091078915	6.4988046060580
15.45455732879956	0	5.9977875629949295	1.3750451644648543	6.7150333332876
18.61604176916242	0	9.775242922861011	1.1886070620237166	4.6575720371267
39.423311410061466	1	13.596888896832859	7.796704003664869	7.0702388780568
30.542003287867175	1	2.0112813125692597	9.02853630401518	9.8384459256866
36.758281614016326	1	19.988865972822232	3.8917486220750854	3.4219600058331
22.38058650336209	1	7.293287714899552	2.595670177298847	4.7848271387979

2. Data Preprocessing

- **Objective**: Clean and prepare the dataset for model building.
- **Steps**:
  - **Load Data**: Import the dataset into a Pandas DataFrame.
  - **Handle Missing Data**: Check for missing values and handle them appropriately (e.g., imputation or removal).
  - **Feature Scaling**: Normalize the features to a common scale (StandardScaler or Min-Max Scaling).
  - **Data Splitting**: Split the data into training and testing sets (typically an 80-20 split)

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm , skew , ttest_ind, f_oneway
from sklearn.preprocessing import LabelEncoder ,  PowerTransformer ,  StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score , precision_score , recall_score , f1_score , roc_auc_score
```

```python
df = pd.read_csv('/content/parkinsons_disease_data.csv')
```

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2105 entries, 0 to 2104
Data columns (total 35 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   PatientID                  2105 non-null   int64
 1   Age                        2105 non-null   int64
 2   Gender                     2105 non-null   int64
 3   Ethnicity                  2105 non-null   int64
 4   EducationLevel             2105 non-null   int64
```

```python
df.describe()
```

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsumption	PhysicalActivity	DietQuality	SleepQuality	FamilyHistoryParkinso
count	2105.000000	2105.000000	2105.000000	2105.000000	2105.000000	2105.000000	2105.000000	2105.000000	2105.000000	2105.000000	2105.000000	2105.0000
mean	4110.000000	69.601900	0.492637	0.692637	1.337292	27.209493	0.296437	10.040413	5.016674	4.912901	6.996639	0.1458
std	607.805479	11.594511	0.500065	1.003827	0.895840	7.208099	0.456795	5.687014	2.890919	2.872115	1.753065	0.3530
min	3058.000000	50.000000	0.000000	0.000000	0.000000	15.008333	0.000000	0.002228	0.004157	0.000011	4.000497	0.0000
25%	3584.000000	60.000000	0.000000	0.000000	1.000000	20.782176	0.000000	5.150278	2.455703	2.478503	5.488864	0.0000
50%	4110.000000	70.000000	0.000000	0.000000	1.000000	27.184571	0.000000	10.070337	5.031550	4.825187	6.929819	0.0000
75%	4636.000000	80.000000	1.000000	1.000000	2.000000	33.462452	1.000000	14.829565	7.512795	7.381487	8.558719	0.0000
max	5162.000000	89.000000	1.000000	3.000000	3.000000	39.999887	1.000000	19.988866	9.995255	9.995864	9.999821	1.0000

```python
df.columns
```

```
Index(['PatientID', 'Age', 'Gender', 'Ethnicity', 'EducationLevel', 'BMI',
       'Smoking', 'AlcoholConsumption', 'PhysicalActivity', 'DietQuality',
       'SleepQuality', 'FamilyHistoryParkinsons', 'TraumaticBrainInjury',
```

```python
df.shape
```

```
(2105, 35)
```

```python
df.head()
```

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsumption	PhysicalActivity	DietQuality	SleepQuality	FamilyHistoryParkinsons	TraumaticBrainInjury
0	3058	85	0	3	1	19.619878	0	5.108241	1.380660	3.893969	9.283194	0	0
1	3059	75	0	0	2	16.247339	1	6.027648	8.409804	8.513428	5.602470	0	0
2	3060	70	1	0	0	15.368239	0	2.242135	0.213275	6.498805	9.929824	0	0
3	3061	52	0	0	0	15.454557	0	5.997788	1.375045	6.715033	4.196189	0	0
4	3062	87	0	0	1	18.616042	0	9.775243	1.188607	4.657572	9.363925	0	0

```python
df.tail()
```

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsumption	PhysicalActivity	DietQuality	SleepQuality	FamilyHistoryParkinsons	TraumaticBrainInju
2100	5158	87	1	0	2	38.483841	0	12.674393	5.325900	5.947278	6.296231	0	
2101	5159	67	0	0	1	33.694396	1	0.977018	0.108599	4.825187	6.342325	1	
2102	5160	65	0	0	2	22.829631	0	6.152286	5.775103	0.334244	9.740019	1	
2103	5161	61	1	0	0	16.871030	1	0.292094	2.280475	9.598513	8.289390	0	
2104	5162	56	0	0	2	16.569934	0	1.985084	1.400441	9.883835	9.930027	0	

```
df.isnull().sum()
```

	0
PatientID	0
Age	0
Gender	0
Ethnicity	0
EducationLevel	0
BMI	0
Smoking	0
AlcoholConsumption	0
PhysicalActivity	0
DietQuality	0
SleepQuality	0
FamilyHistoryParkinsons	0
TraumaticBrainInjury	0
Hypertension	0
Diabetes	0
Depression	0

```
df.drop_duplicates(inplace=True)
```

```
def remove_outliers(df,col):
  q1 = df[col].quantile(0.25)
  q3 = df[col].quantile(0.75)
  iqr = q3-q1
  lower_bound = q1 - 1.5*iqr
  upper_bound = q3 + 1.5*iqr
  df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
  return df
  numerical_columns = [
    'Age', 'BMI', 'SystolicBP', 'DiastolicBP', 'CholesterolTotal',
    'CholesterolLDL', 'CholesterolHDL', 'CholesterolTriglycerides', 'UPDRS',
    'MoCA', 'FunctionalAssessment'
]

data = remove_outliers(df, numerical_columns )
```

```
def check_and_normalize(df, columns):
    pt = PowerTransformer(method='yeo-johnson')
    for col in columns:
        skewness = skew(df[col])
        if abs(skewness) > 0.5:
            df[col] = pt.fit_transform(df[col].values.reshape(-1, 1))
    return df

# Normalize numerical features
data = check_and_normalize(df, numerical_columns)
```

```
df.columns
```

```
Index(['PatientID', 'Age', 'Gender', 'Ethnicity', 'EducationLevel', 'BMI',
       'Smoking', 'AlcoholConsumption', 'PhysicalActivity', 'DietQuality',
       'SleepQuality', 'FamilyHistoryParkinsons', 'TraumaticBrainInjury',
       'Hypertension', 'Diabetes', 'Depression', 'Stroke', 'SystolicBP',
       'DiastolicBP', 'CholesterolTotal', 'CholesterolLDL', 'CholesterolHDL',
       'CholesterolTriglycerides', 'UPDRS', 'MoCA', 'FunctionalAssessment',
       'Tremor', 'Rigidity', 'Bradykinesia', 'PosturalInstability',
       'SpeechProblems', 'SleepDisorders', 'Constipation', 'Diagnosis',
       'DoctorInCharge'],
      dtype='object')
```

```
df.shape
```

```
(2105, 35)
```

```
df.head()
```

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsumption	PhysicalActivity	DietQuality	SleepQuality	FamilyHistoryParkinsons	TraumaticBrainInjury
0	3058	85	0	3	1	19.619878	0	5.108241	1.380660	3.893969	9.283194	0	0
1	3059	75	0	0	2	16.247339	1	6.027648	8.409804	8.513428	5.602470	0	0
2	3060	70	1	0	0	15.368239	0	2.242135	0.213275	6.498805	9.929824	0	0
3	3061	52	0	0	0	15.454557	0	5.997788	1.375045	6.715033	4.196189	0	0
4	3062	87	0	0	1	18.616042	0	9.775243	1.188607	4.657572	9.363925	0	0

```
df.tail()
```

	PatientID	Age	Gender	Ethnicity	EducationLevel	BMI	Smoking	AlcoholConsumption	PhysicalActivity	DietQuality	SleepQuality	FamilyHistoryParkinsons	TraumaticBrainInju
2100	5158	87	1	0	2	38.483841	0	12.674393	5.325900	5.947278	6.296231	0	
2101	5159	67	0	0	1	33.694396	1	0.977018	0.108599	4.825187	6.342325	1	
2102	5160	65	0	0	2	22.829631	0	6.152286	5.775103	0.334244	9.740019	1	
2103	5161	61	1	0	0	16.871030	1	0.292094	2.280475	9.598513	8.289390	0	
2104	5162	56	0	0	2	16.569934	0	1.985084	1.400441	9.883835	9.930037	0	

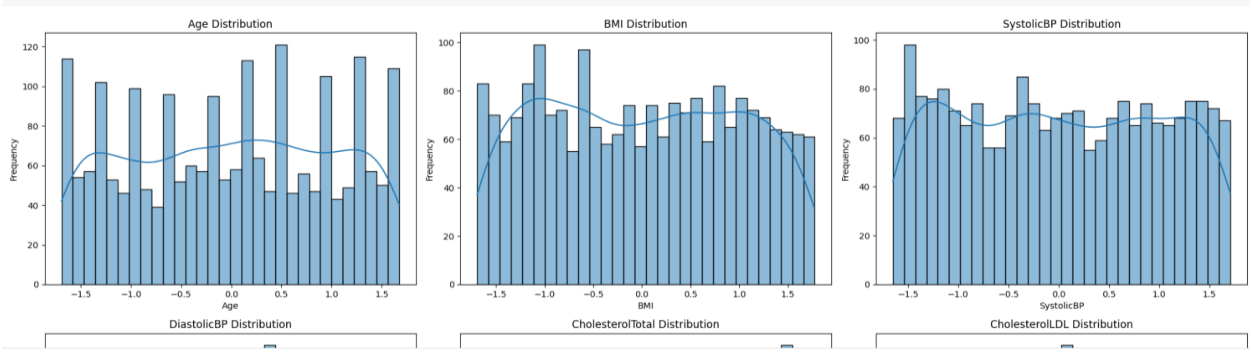
```python
label_encoders = {}
categorical_columns = [
    'Gender', 'Ethnicity', 'EducationLevel', 'Smoking', 'AlcoholConsumption',
    'PhysicalActivity', 'DietQuality', 'SleepQuality', 'FamilyHistoryParkinsons',
    'TraumaticBrainInjury', 'Hypertension', 'Diabetes', 'Depression', 'Stroke',
    'Tremor', 'Rigidity', 'Bradykinesia', 'PosturalInstability', 'SpeechProblems',
    'SleepDisorders', 'Constipation', 'DoctorInCharge'
]

for col in categorical_columns:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le
```

```python
scaler = StandardScaler()
data[numerical_columns] = scaler.fit_transform(data[numerical_columns])
```

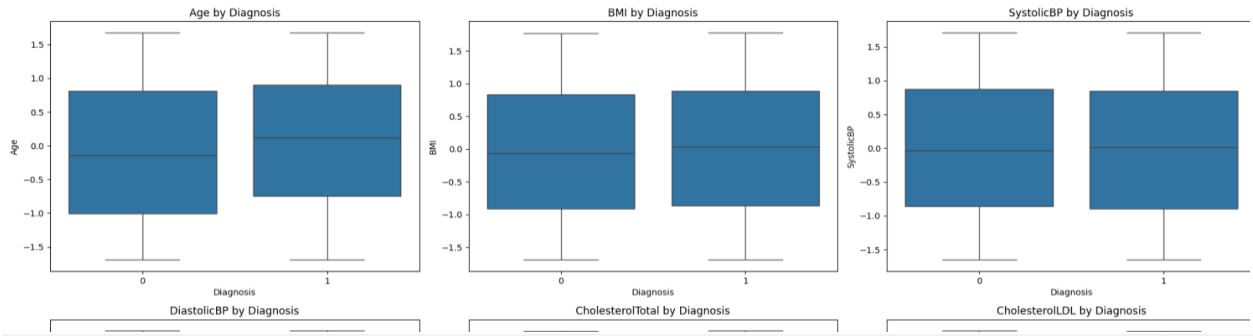
3. Exploratory Data Analysis (EDA)

- **Objective**: Understand the dataset and relationships between features.
- **Steps**:
    - **Visualizations**: Use Seaborn and Matplotlib for histograms, boxplots, and pair plots to visualize the distribution of features.
    - **Correlation Analysis**: Visualize correlations between features to identify potential relationship

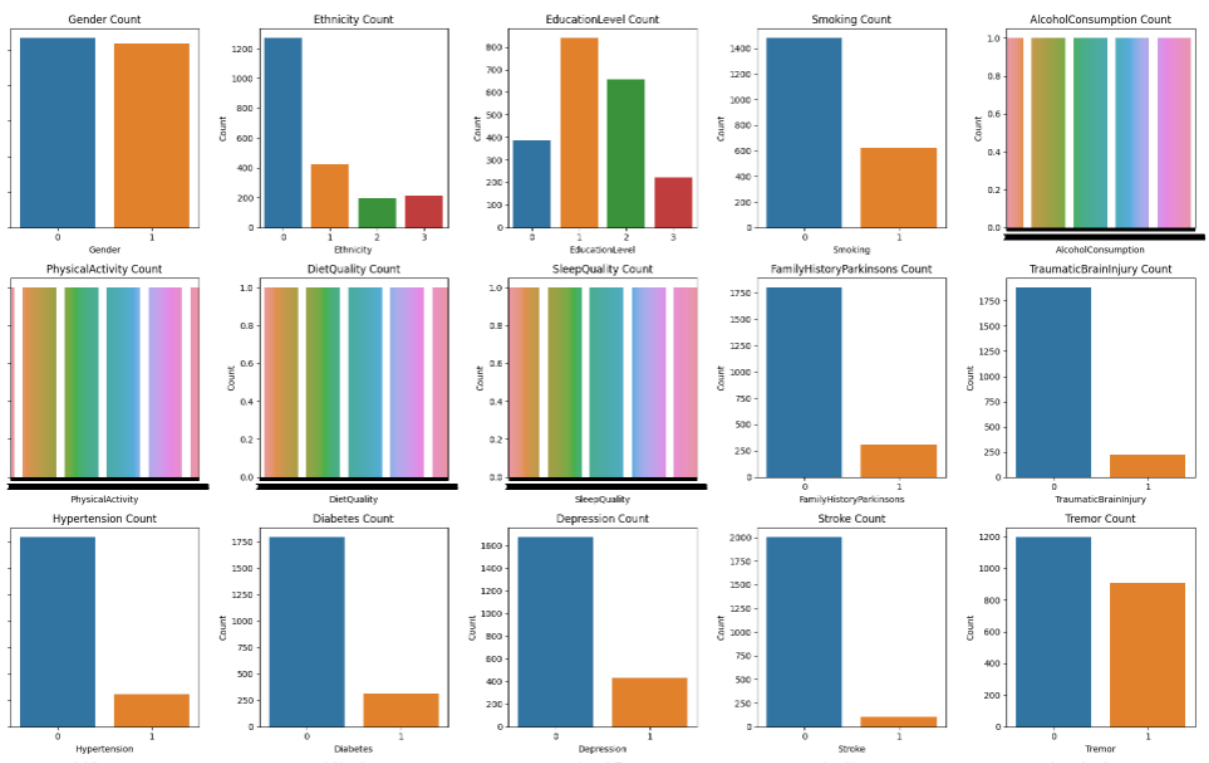
```python
plt.figure(figsize=(20, 20))
for i, col in enumerate(numerical_columns):
    plt.subplot(4, 3, i + 1)
    sns.histplot(data[col], bins=30, kde=True)
    plt.title(f'{col} Distribution')
    plt.xlabel(col)
    plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```



```
plt.figure(figsize=(20, 20))
for i, col in enumerate(numerical_columns):
    plt.subplot(4, 3, i + 1)
    sns.boxplot(x='Diagnosis', y=col, data=data)
    plt.title(f'{col} by Diagnosis')
    plt.xlabel('Diagnosis')
    plt.ylabel(col)
plt.tight_layout()
plt.show()
```




```
categorical_columns = [
    'Gender', 'Ethnicity', 'EducationLevel', 'Smoking', 'AlcoholConsumption',
    'PhysicalActivity', 'DietQuality', 'SleepQuality', 'FamilyHistoryParkinsons',
    'TraumaticBrainInjury', 'Hypertension', 'Diabetes', 'Depression', 'Stroke',
    'Tremor', 'Rigidity', 'Bradykinesia', 'PosturalInstability', 'SpeechProblems',
    'SleepDisorders', 'Constipation', 'DoctorInCharge'
]

plt.figure(figsize=(20, 20))
for i, col in enumerate(categorical_columns):
    plt.subplot(5, 5, i + 1)
    sns.countplot(x=col, data=data)
    plt.title(f'{col} Count')
    plt.xlabel(col)
    plt.ylabel('Count')
plt.tight_layout()
plt.show()
```



4. Model Building

- **Objective**: Train different machine learning models to classify patients.
- **Models Implemented**:
  - **Logistic Regression**: A simple and interpretable linear model.
  - **Support Vector Machine (SVM)**: A robust classifier for high-dimensional data.
  - **Decision Tree**: An ensemble model that uses multiple decision trees.

```
X = data.drop('Diagnosis', axis=1)
y = data['Diagnosis']
```

+ Code + Text

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

```
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Decision Tree': DecisionTreeClassifier(),
    'SVM': SVC(probability=True),
}
```

```
results = []

for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    y_proba = model.predict_proba(X_test)[:, 1]

    results.append({
        'Model': model_name,
        'Accuracy': accuracy_score(y_test, y_pred),
        'Precision': precision_score(y_test, y_pred),
        'Recall': recall_score(y_test, y_pred),
        'F1 Score': f1_score(y_test, y_pred),
        'ROC-AUC': roc_auc_score(y_test, y_proba)
    })

results_df = pd.DataFrame(results)
```

5. Model Evaluation

- **Objective**: Evaluate the performance of each model.
- **Steps**:
  - **Performance Metrics**: Use metrics such as accuracy, precision, recall, and F1 score.
  - **Cross-Validation**: Use k-fold cross-validation to assess how well the models generalize to unseen data.
  - **Confusion Matrix**: Visualize the confusion matrix to assess how well the model is distinguishing between the two classes (Parkinson's vs. Healthy).

Example Accuracy Results:

- **Logistic Regression**: ~81-84%
- **SVM**: ~85-88%
- **Decision Tree**: ~85-88%

```
for col in numerical_columns:
    positive_diagnosis = data[data['Diagnosis'] == 1][col]
    negative_diagnosis = data[data['Diagnosis'] == 0][col]
    t_stat, p_val = ttest_ind(positive_diagnosis, negative_diagnosis)
    print(f'T-Test for {col}: t-statistic = {t_stat}, p-value = {p_val}')
```

```
T-Test for Age: t-statistic = 3.002990356938001, p-value = 0.0027049233136815644
T-Test for BMI: t-statistic = 1.3816123982371822, p-value = 0.1672375010506233
T-Test for SystolicBP: t-statistic = -0.20238841243970515, p-value = 0.8396326632099166
T-Test for DiastolicBP: t-statistic = -1.3338585309523516, p-value = 0.18239465860609355
T-Test for CholesterolTotal: t-statistic = -0.8715141752439457, p-value = 0.38357294939851
994
T-Test for CholesterolLDL: t-statistic = 0.6745059771232261, p-value = 0.5000638258758182
T-Test for CholesterolHDL: t-statistic = -0.9001712750094443, p-value = 0.3681321759912145
T-Test for CholesterolTriglycerides: t-statistic = 0.7159384939942246, p-value = 0.4741088
137454361
T-Test for UPDRS: t-statistic = 19.89569462087449, p-value = 7.64274574620401e-81
T-Test for MoCA: t-statistic = -8.059951795943142, p-value = 1.2668283197650053e-15
T-Test for FunctionalAssessment: t-statistic = -10.591469460824781, p-value = 1.4206933749
950206e-25
```

6. Final Model Selection

- **Objective**: Choose the best model based on the evaluation results.
- **Outcome**: Typically, Decision Tree or SVM will perform best, achieving high accuracy. The selected model is the one with the highest precision and recall, balancing accuracy with minimal false positives and false negatives.

```
results_df = pd.DataFrame(results)
results_df.sort_values(by='ROC-AUC', ascending=False, inplace=True)

results_df
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

	Model	Accuracy	Precision	Recall	F1 Score	ROC-AUC
1	Decision Tree	0.888361	0.921053	0.904059	0.912477	0.882030
0	Logistic Regression	0.790974	0.830325	0.848708	0.839416	0.877196
2	SVM	0.643705	0.643705	1.000000	0.783237	0.397811

7. Conclusion

- The project demonstrates the potential of machine learning to detect Parkinson's disease using voice data. The final model can be deployed to assist healthcare professionals in diagnosing Parkinson's disease early, which can lead to better management and treatment outcomes.