Modelos de Regresión

```
[2] import csv
    import numpy as np
    import matplotlib.pyplot as plt
    from sklearn import linear_model
    from sklearn.model_selection import KFold
    from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
    from sklearn.feature_selection import SelectKBest, r_regression
    from sklearn.feature_selection import SequentialFeatureSelector, RFE

    from sklearn.neighbors import KNeighborsRegressor
    from sklearn.tree import DecisionTreeRegressor
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.neural_network import MLPRegressor
    from sklearn.svm import SVR
```

```
[3] # Archivo CSV
    csv_file = 'parkinsons_updrs.data'
```

```
[4] # Columnas a excluir
    exclude_columns = ['subject#']
    exclude_independent = ['subject#', 'total_UPDRS', 'Jitter(%)', 'Jitter:DDP', 'Shimmer:APQ5', 'HNR']

    # Leer el archivo CSV y preparar los datos
    with open(csv_file, mode='r') as file:
        csv_reader = csv.DictReader(file)

        columns = [column for column in csv_reader.fieldnames if column not in exclude_columns]
        independent = [column for column in csv_reader.fieldnames if column not in exclude_independent]
        dependent = 'total_UPDRS'

        data = {column: [] for column in independent}
        total_UPDRS = []

        for row in csv_reader:
            for column in independent:
                if row[column]:
                    data[column].append(float(row[column]))
                else:
                    data[column].append(None)

            if row[dependent]:
                total_UPDRS.append(float(row[dependent]))
            else:
                total_UPDRS.append(None)

    X = np.array([data[column] for column in independent]).T
    y = np.array(total_UPDRS)
```

```
[17] def filter_method(X, y, feature_names, k=5):
         fselection = SelectKBest(r_regression, k=k)
         fselection.fit(X, y)
         selected_features = fselection.get_support()
         selected_feature_names = np.array(independent)[fselection.get_support()]
         return selected_features, selected_feature_names
```

```
[6] def wrapper_method(X, y, model, feature_names, n_features_to_select='auto', direction='forward'):
        sfs = SequentialFeatureSelector(model, n_features_to_select=n_features_to_select, direction=direction, cv=5)
        sfs.fit(X, y)
        selected_features = sfs.get_support()
        selected_feature_names = np.array(independent)[sfs.get_support()]
        return selected_features, selected_feature_names
```

```
[7] def recursive_method(X, y, model, feature_names, n_features_to_select=5):
        rfe = RFE(model, n_features_to_select=n_features_to_select)
        rfe.fit(X, y)
        selected_features = rfe.get_support()
        selected_feature_names = np.array(independent)[rfe.support_]
        return selected_features, selected_feature_names
```

```
[8] # Función para evaluar el rendimiento del modelo usando validación cruzada
    def evaluate_model(X, y, selected_features, model):
        X_selected = X[:, selected_features]
        kf = KFold(n_splits=5, shuffle=True)

        mse_cv = []
        mae_cv = []
        r2_cv = []

        for train_index, test_index in kf.split(X_selected):
            X_train, X_test = X_selected[train_index], X_selected[test_index]
            y_train, y_test = y[train_index], y[test_index]

            model.fit(X_train, y_train)
            y_pred = model.predict(X_test)

            mse_cv.append(mean_squared_error(y_test, y_pred))
            mae_cv.append(mean_absolute_error(y_test, y_pred))
            r2_cv.append(r2_score(y_test, y_pred))

        return np.mean(mse_cv), np.mean(mae_cv), np.mean(r2_cv)
```

```
[9] def evaluate_with_optimal_features(X, y, selected_features, model):
        X_selected = X[:, selected_features]
        model.fit(X_selected, y)
        y_pred = model.predict(X_selected)

        mse = mean_squared_error(y, y_pred)
        mae = mean_absolute_error(y, y_pred)
        r2 = r2_score(y, y_pred)

        return mse, mae, r2
```

```
print("\nModelo Regresión Lineal")
model = linear_model.LinearRegression()

# Método Filter
print("\nMetodo Filter")
selected_features_filter, selected_feature_names_filter = filter_method(X, y, feature_names=independent, k=5)
print("Predictores óptimos (Filter):", selected_feature_names_filter)
# Primera evaluación
mse_filter, mae_filter, r2_filter = evaluate_model(X, y, selected_features_filter, model)
print(f"Filter Method - MSE: {mse_filter}, MAE: {mae_filter}, R^2: {r2_filter}")
# Segunda evaluación usando las características óptimas seleccionadas
mse_filter_opt, mae_filter_opt, r2_filter_opt = evaluate_with_optimal_features(X, y, selected_features_filter, model)
print(f"OptimalFeatures MSE: {mse_filter_opt}, MAE: {mae_filter_opt}, R^2: {r2_filter_opt}")

# Método Wrapper
print("\nMetodo Wrapper")
selected_features_wrapper, selected_feature_names_wrapper = wrapper_method(X, y, model, feature_names=independent, n_features_to_select=5)
print("Predictores óptimos (Wrapper):", selected_feature_names_wrapper)
# Primera evaluación
mse_wrapper, mae_wrapper, r2_wrapper = evaluate_model(X, y, selected_features_wrapper, model)
print(f"Wrapper Method - MSE: {mse_wrapper}, MAE: {mae_wrapper}, R^2: {r2_wrapper}")
# Segunda evaluación usando las características óptimas seleccionadas
mse_wrapper_opt, mae_wrapper_opt, r2_wrapper_opt = evaluate_with_optimal_features(X, y, selected_features_wrapper, model)
print(f"Optimal Features MSE: {mse_wrapper_opt}, MAE: {mae_wrapper_opt}, R^2: {r2_wrapper_opt}")

# Método Recursivo
print("\nMetodo Recursivo")
selected_features_recursive, selected_feature_names_recursive = recursive_method(X, y, model, feature_names=independent, n_features_to_select=5)
print("Predictores óptimos (Recursivo):", selected_feature_names_recursive)
# Primera evaluación
mse_recursive, mae_recursive, r2_recursive = evaluate_model(X, y, selected_features_recursive, model)
print(f"Recursive Method - MSE: {mse_recursive}, MAE: {mae_recursive}, R^2: {r2_recursive}")
# Segunda evaluación usando las características óptimas seleccionadas
mse_recursive_opt, mae_recursive_opt, r2_recursive_opt = evaluate_with_optimal_features(X, y, selected_features_recursive, model)
print(f"Optimal Features - MSE: {mse_recursive_opt}, MAE: {mae_recursive_opt}, R^2: {r2_recursive_opt}")
```

```
Modelo Regresión Lineal

Metodo Filter
Predictores óptimos (Filter): ['age' 'motor_UPDRS' 'Shimmer:APQ11' 'RPDE' 'PPE']
Filter Method - MSE: 11.203078880535486, MAE: 2.450747170700032, R^2: 0.9019909862716391
OptimalFeatures MSE: 11.184730607028982, MAE: 2.448218768411101, R^2: 0.9022967729109506

Metodo Wrapper
Predictores óptimos (Wrapper): ['sex' 'test_time' 'motor_UPDRS' 'Jitter:RAP' 'Shimmer:DDA']
Wrapper Method - MSE: 11.256079556683371, MAE: 2.436047583103329, R^2: 0.9015480041789216
Optimal Features MSE: 11.234534606172055, MAE: 2.4331868990162913, R^2: 0.9018617144719784

Metodo Recursivo
Predictores óptimos (Recursivo): ['Jitter(Abs)' 'Jitter:RAP' 'Jitter:PPQ5' 'Shimmer:APQ3' 'Shimmer:DDA']
Recursive Method - MSE: 113.87364772968245, MAE: 8.643033954527201, R^2: 0.005023989680104046
Optimal Features - MSE: 113.69703636743088, MAE: 8.63390328026619, R^2: 0.006809573350126774
```

```python
print("\nModelo DecisionTree")
model = DecisionTreeRegressor()

# Método Filter
print("\nMetodo Filter")
selected_features_filter, selected_feature_names_filter = filter_method(X, y, feature_names=independent, k=5)
print("Predictores óptimos (Filter):", selected_feature_names_filter)
# Primera evaluación
mse_filter, mae_filter, r2_filter = evaluate_model(X, y, selected_features_filter, model)
print(f"Filter Method - MSE: {mse_filter}, MAE: {mae_filter}, R^2: {r2_filter}")
# Segunda evaluación usando las características óptimas seleccionadas
mse_filter_opt, mae_filter_opt, r2_filter_opt = evaluate_with_optimal_features(X, y, selected_features_filter, model)
print(f"OptimalFeatures MSE: {mse_filter_opt}, MAE: {mae_filter_opt}, R^2: {r2_filter_opt}")

# Método Wrapper
print("\nMetodo Wrapper")
selected_features_wrapper, selected_feature_names_wrapper = wrapper_method(X, y, model, feature_names=independent, n_features_to_select=5)
print("Predictores óptimos (Wrapper):", selected_feature_names_wrapper)
# Primera evaluación
mse_wrapper, mae_wrapper, r2_wrapper = evaluate_model(X, y, selected_features_wrapper, model)
print(f"Wrapper Method - MSE: {mse_wrapper}, MAE: {mae_wrapper}, R^2: {r2_wrapper}")
# Segunda evaluación usando las características óptimas seleccionadas
mse_wrapper_opt, mae_wrapper_opt, r2_wrapper_opt = evaluate_with_optimal_features(X, y, selected_features_wrapper, model)
print(f"Optimal Features MSE: {mse_wrapper_opt}, MAE: {mae_wrapper_opt}, R^2: {r2_wrapper_opt}")

# Método Recursivo
print("\nMetodo Recursivo")
selected_features_recursive, selected_feature_names_recursive = recursive_method(X, y, model, feature_names=independent, n_features_to_select=5)
print("Predictores óptimos (Recursivo):", selected_feature_names_recursive)
# Primera evaluación
mse_recursive, mae_recursive, r2_recursive = evaluate_model(X, y, selected_features_recursive, model)
print(f"Recursive Method - MSE: {mse_recursive}, MAE: {mae_recursive}, R^2: {r2_recursive}")
# Segunda evaluación usando las características óptimas seleccionadas
mse_recursive_opt, mae_recursive_opt, r2_recursive_opt = evaluate_with_optimal_features(X, y, selected_features_recursive, model)
print(f"Optimal Features - MSE: {mse_recursive_opt}, MAE: {mae_recursive_opt}, R^2: {r2_recursive_opt}")
```

```
Modelo DecisionTree

Metodo Filter
Predictores óptimos (Filter): ['age' 'motor_UPDRS' 'Shimmer:APQ11' 'RPDE' 'PPE']
Filter Method - MSE: 1.1288488519829787, MAE: 0.3551130212765958З, R^2: 0.9901153673338328
OptimalFeatures MSE: 2.3441589248252635e-30, MAE: 4.456765925576118e-16, R^2: 1.0

Metodo Wrapper
Predictores óptimos (Wrapper): ['sex' 'test_time' 'motor_UPDRS' 'Jitter:PPQ5' 'Shimmer']
Wrapper Method - MSE: 2.076702392771064, MAE: 0.3123425191489363, R^2: 0.9818483809640629
Optimal Features MSE: 2.7656456280573923e-30, MAE: 5.14160817940447e-16, R^2: 1.0

Metodo Recursivo
Predictores óptimos (Recursivo): ['age' 'sex' 'test_time' 'motor_UPDRS' 'DFA']
Recursive Method - MSE: 0.1498962853940426, MAE: 0.041751540425532044, R^2: 0.998690459275543
Optimal Features - MSE: 3.358466206193693e-30, MAE: 6.391861035731285e-16, R^2: 1.0
```

```python
[13]
print("\nModelo RandomForest")
model = RandomForestRegressor(n_estimators=100)

# Método Filter
print("\nMetodo Filter")
selected_features_filter, selected_feature_names_filter = filter_method(X, y, feature_names=independent, k=5)
print("Predictores óptimos (Filter):", selected_feature_names_filter)
# Primera evaluación
mse_filter, mae_filter, r2_filter = evaluate_model(X, y, selected_features_filter, model)
print(f"Filter Method - MSE: {mse_filter}, MAE: {mae_filter}, R^2: {r2_filter}")
# Segunda evaluación usando las características óptimas seleccionadas
mse_filter_opt, mae_filter_opt, r2_filter_opt = evaluate_with_optimal_features(X, y, selected_features_filter, model)
print(f"OptimalFeatures MSE: {mse_filter_opt}, MAE: {mae_filter_opt}, R^2: {r2_filter_opt}")

# Método Wrapper
print("\nMetodo Wrapper")
selected_features_wrapper, selected_feature_names_wrapper = wrapper_method(X, y, model, feature_names=independent, n_features_to_select=5)
print("Predictores óptimos (Wrapper):", selected_feature_names_wrapper)
# Primera evaluación
mse_wrapper, mae_wrapper, r2_wrapper = evaluate_model(X, y, selected_features_wrapper, model)
print(f"Wrapper Method - MSE: {mse_wrapper}, MAE: {mae_wrapper}, R^2: {r2_wrapper}")
# Segunda evaluación usando las características óptimas seleccionadas
mse_wrapper_opt, mae_wrapper_opt, r2_wrapper_opt = evaluate_with_optimal_features(X, y, selected_features_wrapper, model)
print(f"Optimal Features MSE: {mse_wrapper_opt}, MAE: {mae_wrapper_opt}, R^2: {r2_wrapper_opt}")

# Método Recursivo
print("\nMetodo Recursivo")
selected_features_recursive, selected_feature_names_recursive = recursive_method(X, y, model, feature_names=independent, n_features_to_select=5)
print("Predictores óptimos (Recursivo):", selected_feature_names_recursive)
# Primera evaluación
mse_recursive, mae_recursive, r2_recursive = evaluate_model(X, y, selected_features_recursive, model)
print(f"Recursive Method - MSE: {mse_recursive}, MAE: {mae_recursive}, R^2: {r2_recursive}")
# Segunda evaluación usando las características óptimas seleccionadas
mse_recursive_opt, mae_recursive_opt, r2_recursive_opt = evaluate_with_optimal_features(X, y, selected_features_recursive, model)
print(f"Optimal Features - MSE: {mse_recursive_opt}, MAE: {mae_recursive_opt}, R^2: {r2_recursive_opt}")
```

```
Modelo RandomForest

Metodo Filter
Predictores óptimos (Filter): ['age' 'motor_UPDRS' 'Shimmer:APQ11' 'RPDE' 'PPE']
Filter Method - MSE: 0.6837927679891348, MAE: 0.450980964425532, R^2: 0.9940189035296536
OptimalFeatures MSE: 0.07672788389307474, MAE: 0.139107479659575, R^2: 0.9993297503420103

Metodo Wrapper
Predictores óptimos (Wrapper): ['motor_UPDRS' 'Jitter(Abs)' 'Jitter:PPQ5' 'RPDE' 'DFA']
Wrapper Method - MSE: 3.759719358458436, MAE: 1.232757962042553, R^2: 0.9670870819155404
Optimal Features MSE: 0.48379026732291197, MAE: 0.43083481072340435, R^2: 0.9957738928175868

Metodo Recursivo
Predictores óptimos (Recursivo): ['age' 'sex' 'test_time' 'motor_UPDRS' 'DFA']
Recursive Method - MSE: 0.04868141092181374, MAE: 0.0827340350638309, R^2: 0.9995752638230542
Optimal Features - MSE: 0.0047230285202215375, MAE: 0.02021453106383698, R^2: 0.9999587424012011
```

```python
[16]
print("\nModelo Support Vector Machines")
model = SVR(kernel='rbf')

# Método Filter
print("\nMetodo Filter")
selected_features_filter, selected_feature_names_filter = filter_method(X, y, feature_names=independent, k=5)
print("Predictores óptimos (Filter):", selected_feature_names_filter)
# Primera evaluación
mse_filter, mae_filter, r2_filter = evaluate_model(X, y, selected_features_filter, model)
print(f"Filter Method - MSE: {mse_filter}, MAE: {mae_filter}, R^2: {r2_filter}")
# Segunda evaluación usando las características óptimas seleccionadas
mse_filter_opt, mae_filter_opt, r2_filter_opt = evaluate_with_optimal_features(X, y, selected_features_filter, model)
print(f"OptimalFeatures MSE: {mse_filter_opt}, MAE: {mae_filter_opt}, R^2: {r2_filter_opt}")

# Método Wrapper
print("\nMetodo Wrapper")
selected_features_wrapper, selected_feature_names_wrapper = wrapper_method(X, y, model, feature_names=independent, n_features_to_select=5)
print("Predictores óptimos (Wrapper):", selected_feature_names_wrapper)
# Primera evaluación
mse_wrapper, mae_wrapper, r2_wrapper = evaluate_model(X, y, selected_features_wrapper, model)
print(f"Wrapper Method - MSE: {mse_wrapper}, MAE: {mae_wrapper}, R^2: {r2_wrapper}")
# Segunda evaluación usando las características óptimas seleccionadas
mse_wrapper_opt, mae_wrapper_opt, r2_wrapper_opt = evaluate_with_optimal_features(X, y, selected_features_wrapper, model)
print(f"Optimal Features MSE: {mse_wrapper_opt}, MAE: {mae_wrapper_opt}, R^2: {r2_wrapper_opt}")
```

```
Modelo Support Vector Machines

Metodo Filter
Predictores óptimos (Filter): ['age' 'motor_UPDRS' 'Shimmer:APQ11' 'RPDE' 'PPE']
Filter Method - MSE: 12.095384238076184, MAE: 2.4768780492418285, R^2: 0.8940408597520804
OptimalFeatures MSE: 12.068029169584687, MAE: 2.472023613334542, R^2: 0.8945807962748593

Metodo Wrapper
Predictores óptimos (Wrapper): ['sex' 'motor_UPDRS' 'RPDE' 'DFA' 'PPE']
Wrapper Method - MSE: 11.221245214310715, MAE: 2.3698793498035, R^2: 0.9020594606139847
Optimal Features MSE: 11.089456849694683, MAE: 2.351285224844805, R^2: 0.9031290284095864
```

```python
print("\nModelo NeuralNetwork Regressor")
model = MLPRegressor(hidden_layer_sizes=(1000,), max_iter=10000)

# Método Filter
print("\nMetodo Filter")
selected_features_filter, selected_feature_names_filter = filter_method(X, y, feature_names=independent, k=5)
print("Predictores óptimos (Filter):", selected_feature_names_filter)
# Primera evaluación
mse_filter, mae_filter, r2_filter = evaluate_model(X, y, selected_features_filter, model)
print(f"Filter Method - MSE: {mse_filter}, MAE: {mae_filter}, R^2: {r2_filter}")
# Segunda evaluación usando las características óptimas seleccionadas
mse_filter_opt, mae_filter_opt, r2_filter_opt = evaluate_with_optimal_features(X, y, selected_features_filter, model)
print(f"OptimalFeatures MSE: {mse_filter_opt}, MAE: {mae_filter_opt}, R^2: {r2_filter_opt}")

# Método Wrapper
print("\nMetodo Wrapper")
selected_features_wrapper, selected_feature_names_wrapper = wrapper_method(X, y, model, feature_names=independent, n_features_to_select=5)
print("Predictores óptimos (Wrapper):", selected_feature_names_wrapper)
# Primera evaluación
mse_wrapper, mae_wrapper, r2_wrapper = evaluate_model(X, y, selected_features_wrapper, model)
print(f"Wrapper Method - MSE: {mse_wrapper}, MAE: {mae_wrapper}, R^2: {r2_wrapper}")
# Segunda evaluación usando las características óptimas seleccionadas
mse_wrapper_opt, mae_wrapper_opt, r2_wrapper_opt = evaluate_with_optimal_features(X, y, selected_features_wrapper, model)
print(f"Optimal Features MSE: {mse_wrapper_opt}, MAE: {mae_wrapper_opt}, R^2: {r2_wrapper_opt}")

# Método Recursivo
print("\nMetodo Recursivo")
selected_features_recursive, selected_feature_names_recursive = recursive_method(X, y, model, feature_names=independent, n_features_to_select=5)
print("Predictores óptimos (Recursivo):", selected_feature_names_recursive)
# Primera evaluación
mse_recursive, mae_recursive, r2_recursive = evaluate_model(X, y, selected_features_recursive, model)
print(f"Recursive Method - MSE: {mse_recursive}, MAE: {mae_recursive}, R^2: {r2_recursive}")
# Segunda evaluación usando las características óptimas seleccionadas
mse_recursive_opt, mae_recursive_opt, r2_recursive_opt = evaluate_with_optimal_features(X, y, selected_features_recursive, model)
print(f"Optimal Features - MSE: {mse_recursive_opt}, MAE: {mae_recursive_opt}, R^2: {r2_recursive_opt}")
```

```
Modelo NeuralNetwork Regressor

Metodo Filter
Predictores óptimos (Filter): ['age' 'motor_UPDRS' 'Shimmer:APQ11' 'RPDE' 'PPE']
Filter Method - MSE: 11.513498629847252, MAE: 2.485760263472933, R^2: 0.8993740500559412
OptimalFeatures MSE: 10.775593719326158, MAE: 2.4392900012722083, R^2: 0.9058707520843617
```

1. **Consideras que el modelo de regresión lineal es adecuado para los datos. ¿Por qué?**

   El mejor R^2 obtenido con regresión lineal fue el de Filter, con 0.9022, ya ajustando el modelo con las optimal features. Este en mi opinión es un resultado aceptable, sin embargo con un valor de 11.18 para el MMSE, y 2.45 de MAE, está lejos de ser el mejor modelo para los datos.

2. **¿Qué método de selección de características consideras que funciona bien con los datos? ¿Por qué?**

   El método de RandomForest (Recursivo) dio uno de los mejores resultados con un MSE de 0.004 y un MAE de 0.02.

   Sin embargo el método de DecisionTree (Recursivo) fue el que ha dado los mejores resultados con valores de MSE y MAE elevados a la -30 y -16 respectivamente, dando por mucho el mejor margen de error, y r cuadrada.

3. **Del proceso de selección de características, ¿puedes identificar algunas que sean sobresalientes? ¿Qué información relevantes observas de dichas características?**

   De las características seleccionadas, age, sex y test_time fueron las que más se repitieron, con Shimmer:APQ11, motor_UPDRS y PPE repitiendose menos pero de igual manera aparecieron en varios modelos. Lo cual me hace pensar que son las más importantes en ese orden.

4. **¿Los modelos de regresión no lineal funcionaron mejor que el lineal? ¿Por qué?**

   En este caso nuestro conjunto de datos probablemente tiene relaciones entre varias variables, por lo que un modelo lineal no es lo suficientemente preciso.

5. **¿Se puede concluir algo interesante sobre los resultados de modelar estos datos con regresión?**

   Todo apunta a que la regresión lineal no es un buen candidato para realizar el análisis de nuestros datos. Las redes neuronales, siendo el último modelo que probé, requieren de un tiempo de procesamiento exagerado y al menos con parametros de hidden_layer_sizes=(1000) y max_iter=10000 no ofrece una buena predicción a comparación de modelos como random forest y decision tree.