API de los Diamantes.

Dataset

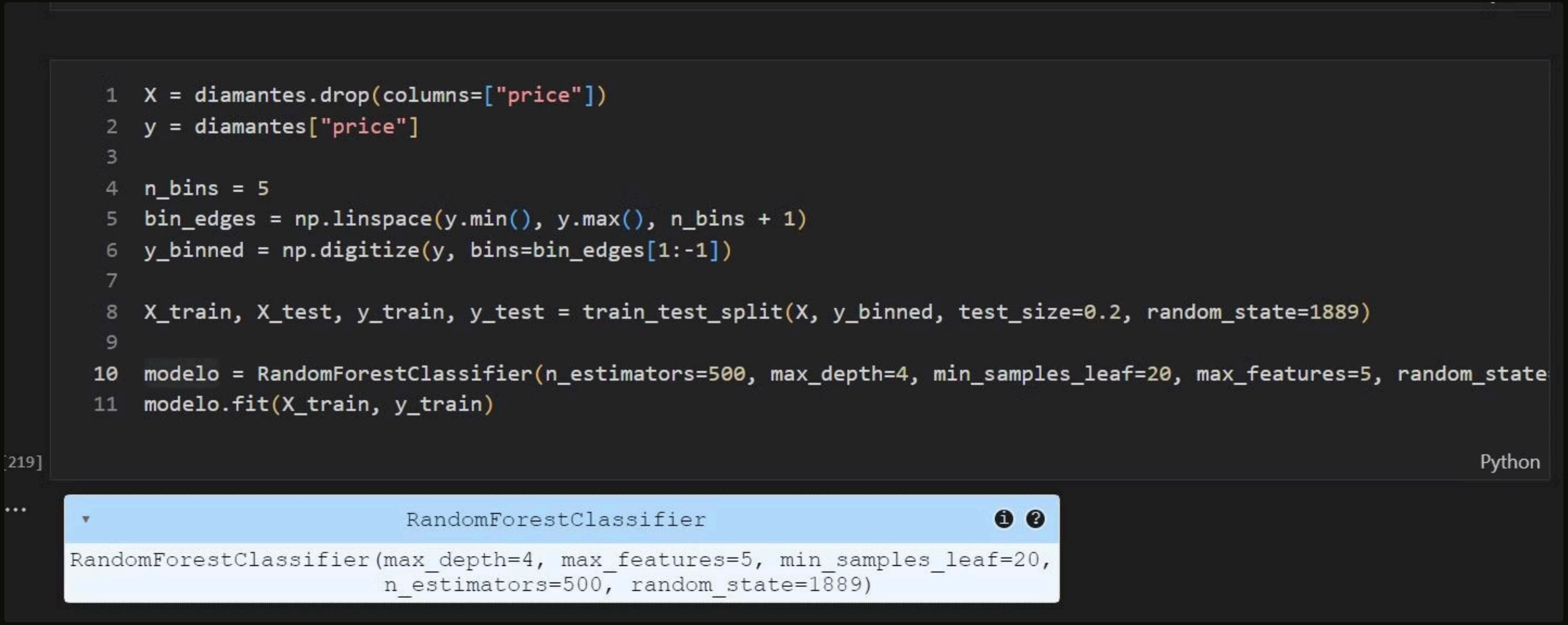
Train

Test

Predicciones

Una query..o dos

Modelo



```python
X = diamantes.drop(columns=["price"])
y = diamantes["price"]

n_bins = 5
bin_edges = np.linspace(y.min(), y.max(), n_bins + 1)
y_binned = np.digitize(y, bins=bin_edges[1:-1])

X_train, X_test, y_train, y_test = train_test_split(X, y_binned, test_size=0.2, random_state=1889)

modelo = RandomForestClassifier(n_estimators=500, max_depth=4, min_samples_leaf=20, max_features=5, random_state
modelo.fit(X_train, y_train)
```

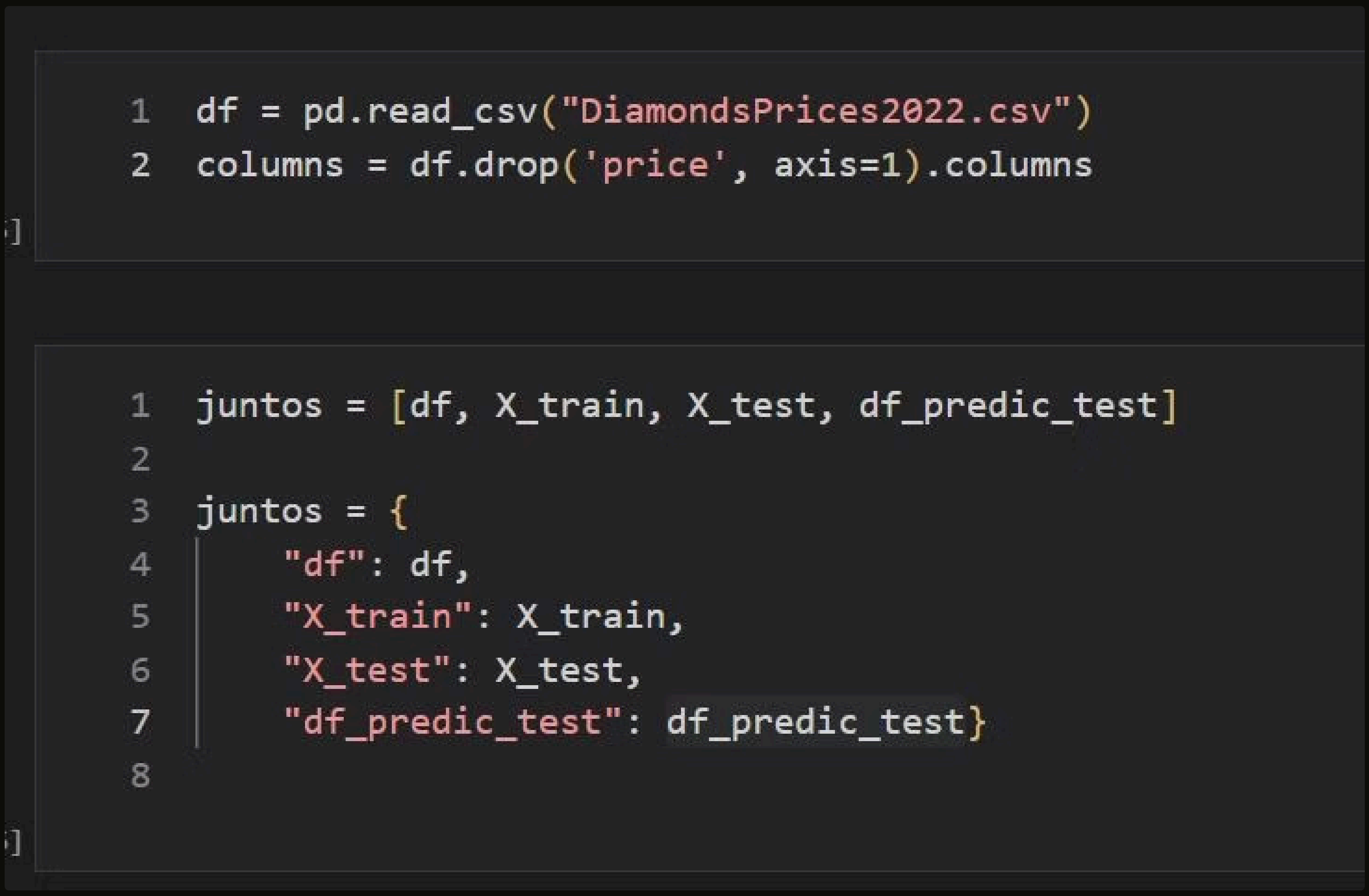
[219]

Python

```
RandomForestClassifier
RandomForestClassifier(max_depth=4, max_features=5, min_samples_leaf=20,
                       n_estimators=500, random_state=1889)
```

Made with Gamma

Union del data frame, train, test, predicciones

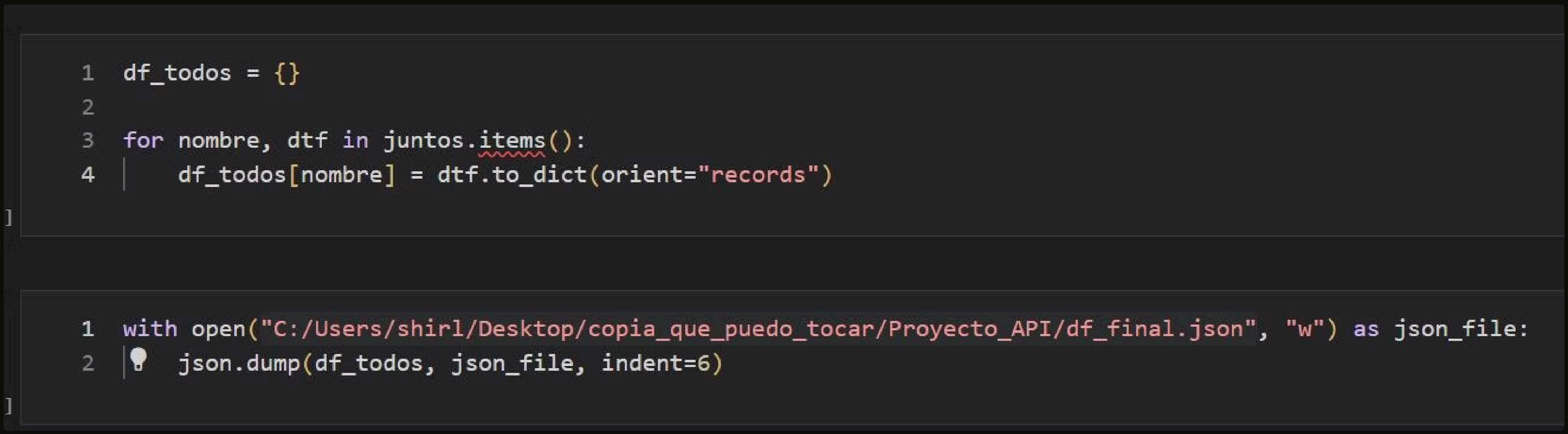


```python
df = pd.read_csv("DiamondsPrices2022.csv")
columns = df.drop('price', axis=1).columns
```

```python
juntos = [df, X_train, X_test, df_predic_test]

juntos = {
    "df": df,
    "X_train": X_train,
    "X_test": X_test,
    "df_predic_test": df_predic_test}

```

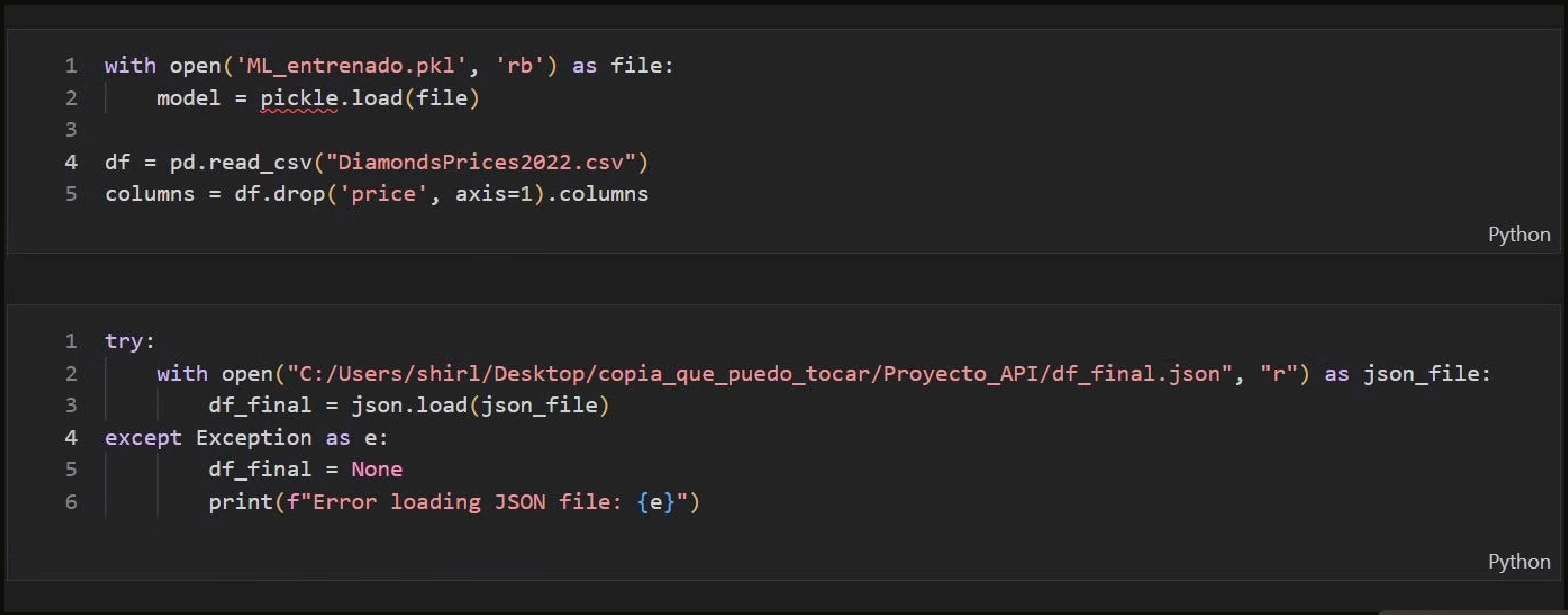
Se creo que data frame en json.

Modelo en un pickle.



```python
df_todos = {}

for nombre, dtf in juntos.items():
    df_todos[nombre] = dtf.to_dict(orient="records")
```

```python
with open("C:/Users/shirl/Desktop/copia_que_puedo_tocar/Proyecto_API/df_final.json", "w") as json_file:
    json.dump(df_todos, json_file, indent=6)
```



```python
with open('ML_entrenado.pkl', 'rb') as file:
    model = pickle.load(file)

df = pd.read_csv("DiamondsPrices2022.csv")
columns = df.drop('price', axis=1).columns
```

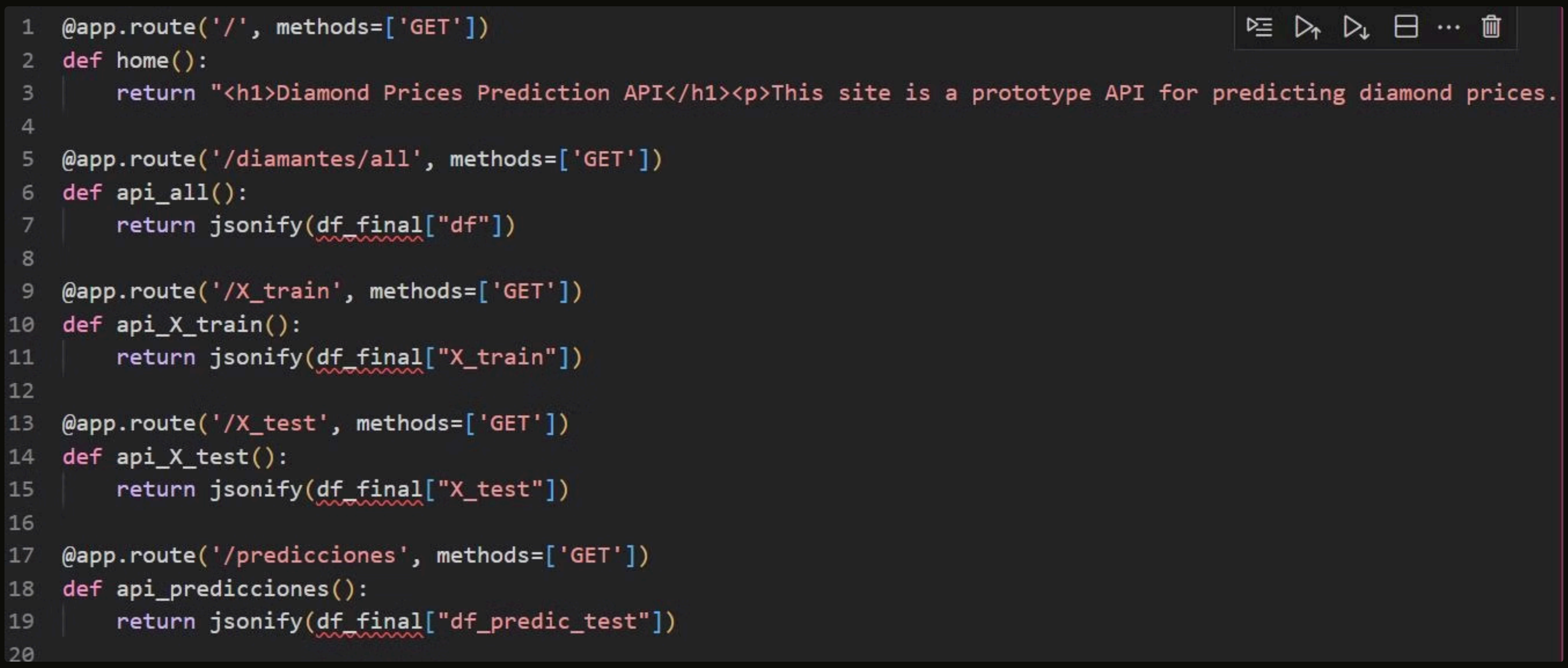
Python

```python
try:
    with open("C:/Users/shirl/Desktop/copia_que_puedo_tocar/Proyecto_API/df_final.json", "r") as json_file:
        df_final = json.load(json_file)
except Exception as e:
        df_final = None
        print(f"Error loading JSON file: {e}")
```

Python

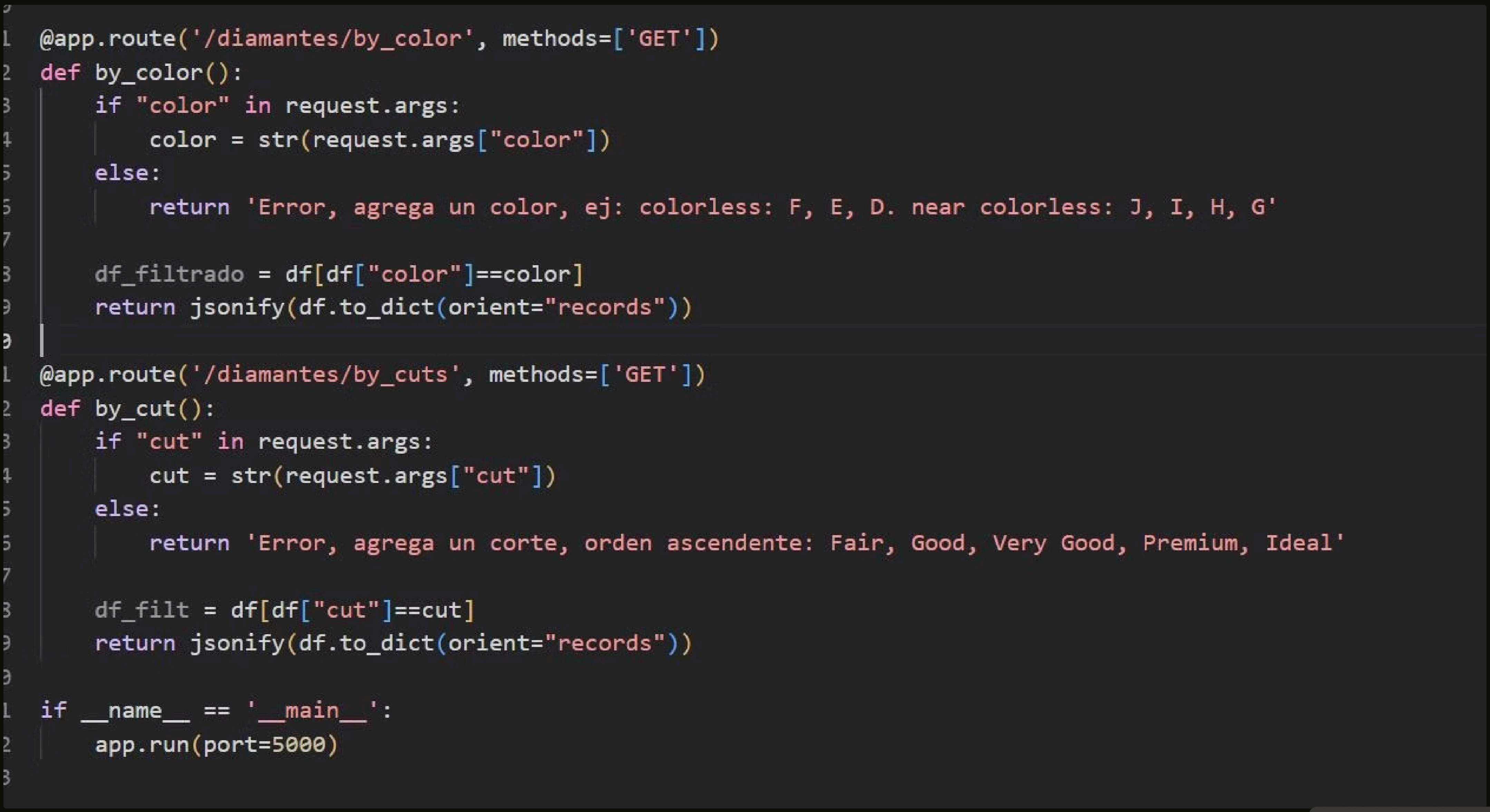
Rutas

DF, Train, Test, Predicciones



```python
@app.route('/', methods=['GET'])
def home():
    return "<h1>Diamond Prices Prediction API</h1><p>This site is a prototype API for predicting diamond prices.

@app.route('/diamantes/all', methods=['GET'])
def api_all():
    return jsonify(df_final["df"])

@app.route('/X_train', methods=['GET'])
def api_X_train():
    return jsonify(df_final["X_train"])

@app.route('/X_test', methods=['GET'])
def api_X_test():
    return jsonify(df_final["X_test"])

@app.route('/predicciones', methods=['GET'])
def api_predicciones():
    return jsonify(df_final["df_predic_test"])
```

Filtrando por color, corte



```python
@app.route('/diamantes/by_color', methods=['GET'])
def by_color():
    if "color" in request.args:
        color = str(request.args["color"])
    else:
        return 'Error, agrega un color, ej: colorless: F, E, D. near colorless: J, I, H, G'

    df_filtrado = df[df["color"]==color]
    return jsonify(df.to_dict(orient="records"))

@app.route('/diamantes/by_cuts', methods=['GET'])
def by_cut():
    if "cut" in request.args:
        cut = str(request.args["cut"])
    else:
        return 'Error, agrega un corte, orden ascendente: Fair, Good, Very Good, Premium, Ideal'

    df_filt = df[df["cut"]==cut]
    return jsonify(df.to_dict(orient="records"))

if __name__ == '__main__':
    app.run(port=5000)
```

Ahora en acción…