In [1]:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
import joblib
from flask import Flask, request, jsonify, render_template
import pickle
```

In [2]:

```
#Load the csv file
data = pd.read_csv("/Users/srilathasirigala/Documents/Intern/Kerala_Loksabha_1962_2019(1).csv")
#data = pd.get_dummies(data, columns=["Ambalapuzha"])
#Alternatively, you can use scikit-learn's LabelEncoder to encode categorical variables as integer values. For example:
data.info()
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 296 entries, 0 to 295
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   PC_Name               296 non-null    object
 1   No                    296 non-null    int64
 2   Type                  296 non-null    object
 3   State                 296 non-null    object
 4   Winning_candidate     296 non-null    object
 5   Party                 296 non-null    object
 6   Electors              296 non-null    int64
 7   Vote                  296 non-null    int64
 8   Turnout               296 non-null    float64
 9   Margin                296 non-null    int64
 10  Margin_in_percentage  296 non-null    float64
 11  year                  296 non-null    int64
dtypes: float64(2), int64(5), object(5)
memory usage: 27.9+ KB
```

Out[2]:

| | PC_Name | No | Type | State | Winning_candidate | Party | Electors | Vote | Turnout | Margin | Margin_in_percentage | year |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Ambalapuzha | 143 | GEN | Kerala | P. K. Vasudevan Nair | Communist Party Of India | 445802 | 334846 | 75.1 | 11233 | 3.4 | 1962 |
| **1** | Badagara | 133 | GEN | Kerala | A. V. Raghavan | Independent | 463498 | 343312 | 74.1 | 72907 | 21.2 | 1962 |
| **2** | Chirayinkil | 147 | GEN | Kerala | M. K. Kumaran | Communist Party Of India | 437189 | 311762 | 71.3 | 33219 | 10.7 | 1962 |
| **3** | Ernakulam | 140 | GEN | Kerala | A. M. Thomas | Indian National Congress | 455280 | 363493 | 79.8 | 23399 | 6.4 | 1962 |
| **4** | Kasergod | 131 | GEN | Kerala | A. K. Gopalan | Communist Party Of India | 460358 | 308449 | 67.0 | 83363 | 27.0 | 1962 |

In [3]:

```
data.isnull().sum()
```

Out[3]:

```
PC_Name                 0
No                      0
Type                    0
State                   0
Winning_candidate       0
Party                   0
Electors                0
Vote                    0
Turnout                 0
Margin                  0
Margin_in_percentage    0
year                    0
dtype: int64
```

In [4]:

```
data.describe()
```

Out[4]:

| | No | Electors | Vote | Turnout | Margin | Margin_in_percentage | year |
|---|---|---|---|---|---|---|---|
| **count** | 296.000000 | 2.960000e+02 | 2.960000e+02 | 296.000000 | 296.000000 | 296.000000 | 296.000000 |
| **mean** | 58.334459 | 8.808178e+05 | 6.457784e+05 | 73.026689 | 52024.831081 | 8.952365 | 1991.006757 |
| **std** | 121.482787 | 2.743997e+05 | 2.172038e+05 | 7.054053 | 46519.670105 | 8.024775 | 16.463826 |
| **min** | 1.000000 | 4.096620e+05 | 2.111360e+05 | 45.800000 | 529.000000 | 0.100000 | 1962.000000 |
| **25%** | 7.000000 | 6.072050e+05 | 4.309175e+05 | 69.200000 | 18641.500000 | 2.875000 | 1977.000000 |
| **50%** | 13.000000 | 9.743710e+05 | 7.095725e+05 | 73.800000 | 42279.000000 | 6.800000 | 1991.000000 |
| **75%** | 19.000000 | 1.101156e+06 | 7.946598e+05 | 77.750000 | 72684.250000 | 13.050000 | 2004.000000 |
| **max** | 496.000000 | 1.332683e+06 | 1.100051e+06 | 88.500000 | 431770.000000 | 46.700000 | 2019.000000 |

In [5]:

```
#Select the independent and dependent variables
X=data[['PC_Name','No','Type','State','Winning_candidate','Electors','Vote','Turnout','Margin','Margin_in_percentage']]
y=data['Party']
```

In [6]:

```
# perform one-hot encoding on the categorical features
X = pd.get_dummies(X)
```

In [7]:

```
#split the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [8]:

```
#feature Scaling
Sc=StandardScaler()
X_train=Sc.fit_transform(X_train)
X_test=Sc.transform(X_test)
```

In [9]:

```
from sklearn.ensemble import RandomForestClassifier

classifier=RandomForestClassifier()
```

In [10]:

```
pickle.dump(classifier,open("model2.pkl",'wb'))
```

In [ ]:

```
#Save the trained model to a file

# Define the Flask app

app = Flask(__name__,template_folder='/Users/srilathasirigala/Documents/Intern/MLModelDeployment/Templates')
modele=pickle.load(open("model2.pkl",'rb'))

# Define the API endpoint for making predictions
@app.route("/")
def Home():
    return render_template("index.html")

@app.route("/predict", methods=["POST"])
def predict():
    # Get the input features from the request
    data = request.get_json()
    features = [data["feature1"], data["feature2"], data["feature3"], data["feature4"],data["feature5"], data["feature6"], data["feature 7"], data["feature8"],data["fea

    # Make a prediction with the model
    prediction = modele.predict([features])[0]

    # Return the prediction as a JSON object
    response = {"prediction": prediction}
    return render_template('index1.html',prediction_text="Kerala_Loksabha_1962_2019".format(prediction))

# Start the app
if __name__ == "__main__":
    app.run(debug=True,port=5002, use_reloader=False)
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
```

```
 * Running on http://127.0.0.1:5002/ (Press CTRL+C to quit)
127.0.0.1 - - [04/Apr/2023 15:46:04] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [04/Apr/2023 15:46:04] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [04/Apr/2023 15:53:02] "GET / HTTP/1.1" 200 -
```

In [ ]: