US Presidential Election Model

Francesco Favagrossa

July 2024

1 Introduction

US Presidential election are just around the corner. Several political scientists and Economists have been trying their best to produce the most accurate model to forecast the result of the 2024 US Presidential election. However looking at the most cited models I noticed that some explanatory variables I consider crucial were not taken into account. Moreover I noticed a big difference between the forecast produced by the official models and the ones produced by the the bets market. The Youtube Channel "Rational Economics" made a video about the bets market on the Presidential Election. Moved by these aspects I decided to develop my own Model.

2 The Model

As i said in the introduction there are some explainatory variables i considers crucial that were not taken into account by the officials model. The model is an old fashioned Probit Regression. The main difference between the model I developed and the officials ones lies on the fact that I decided to take a Micro approach, hence instead of looking at aggregate data for the United States as a whole a looked at State-by-State level data. The variables I decided to take into account to forecast the Presidential Election are:

- Gross Domestic Product Annual Growth: X_1
- Year on Year Inflation: X_2
- The Level of Average House Price: X_3
- The share of the Population holding a Degree: X_4
- The Level of Per Capita Personal Income: X_5

Here the model:

$$P(Y = 1|X) = \Phi(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5)$$

where:

- Y is the binary dependent variable, in the model 0 represents Democrats and 1 Republicans
- Φ is the cumulative distribution function of the standard normal distribution.
- β_0 is the intercept.
- $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$ are the coefficients for the regressors.
- X_1, X_2, X_3, X_4, X_5 are the independent variables.

In matrix form, this can be written as:

$$P(Y = 1|X) = \Phi(\mathbf{X}\boldsymbol{\beta})$$

where:

- $\mathbf{X} = [1, X_1, X_2, X_3, X_4, X_5]$ is the vector of independent variables including the intercept.
- $\boldsymbol{\beta} = [\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5]^T$ is the vector of coefficients.

3 Estimation & Methodology

The process begins by loading and preprocessing the data, followed by model training and prediction. The main steps involved are:

1. **Data Loading**: The historical economic data is loaded from an Excel file, which contains multiple sheets, each representing different datasets. The sheets are read one by one.
2. **Feature Selection and Scaling**: Key economic indicators (GDP growth, inflation, house prices, college graduates, and personal income) are selected as features for the model. These features are scaled using `StandardScaler` to normalize the data.
3. **Model Training**: A Probit regression model is trained using the preprocessed data. The model uses stochastic gradient descent (SGD) for optimization, and binary cross-entropy loss as the criterion.
4. **Prediction**: After training, the model is used to predict election results on new datasets.

3.1 Implementation

Below is a detailed explanation of the code used in the implementation:

3.2 Iterate Over Sheets

```
for sheet_name in train_excel_data.sheet_names:
    ...
```

The code iterates over each sheet in the training Excel file, processing each dataset individually.

3.3 Load Training Data

```
train_data = pd.read_excel(train_file_path, sheet_name=sheet_name)
```

This line reads the data from the current sheet into a pandas DataFrame.

3.4 Feature Selection and Scaling

```
X = train_data[['GDP_G', 'Inflation', 'House_price',
    'College_Graduates', 'Personal_Income']].values

y = train_data['United_States'].values

scaler = StandardScaler()

X = scaler.fit_transform(X)
```

Here, the selected features are extracted, and the target variable (`United_States`) is isolated. The features are then scaled to have zero mean and unit variance.

3.5 Convert Data to Tensors

```
X = torch.tensor(X, dtype=torch.float32)
y = torch.tensor(y, dtype=torch.float32).view(-1, 1)
```

The scaled features and target variable are converted to PyTorch tensors to be used in the Probit regression model.

3.6 Model Initialization

```
model = ProbitRegression(X.shape[1])
criterion = nn.BCELoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)
```

A Probit regression model is initialized with the appropriate input size. The binary cross-entropy loss function and SGD optimizer are set up for training.

3.7 Model Training

The model is trained over a significant number of epochs to ensure convergence and optimal performance. Here are the detailed steps involved in the training process:

```
num_epochs = 1000
for epoch in range(num_epochs):
    model.train()
    optimizer.zero_grad()        # Clear gradients from the previous step
    outputs = model(X)           # Forward pass: compute predicted outputs
    loss = criterion(outputs, y) # Compute the loss
    loss.backward()              # Backward pass: compute gradients
    optimizer.step()             # Update model parameters
```

During each epoch:

- `model.train()` sets the model to training mode.

- `optimizer.zero_grad()` clears the gradients from the previous step to prevent accumulation.
- `outputs = model(X)` performs a forward pass to compute the predicted outputs given the input data X.
- `loss = criterion(outputs, y)` calculates the loss between the predicted outputs and the true target values y using binary cross-entropy loss.
- `loss.backward()` computes the gradients of the loss with respect to the model parameters.
- `optimizer.step()` updates the model parameters using the computed gradients.

3.8 Load and Scale New Data for Prediction

```
predict_excel_data = pd.read_excel(predict_file_path, sheet_name=sheet_name)

X_new = predict_excel_data[['GDP_G', 'Inflation', 'House_price',
        'College_Graduates', 'Personal_Income']].values

X_new = scaler.fit_transform(X_new)

X_new = torch.tensor(X_new, dtype=torch.float32)
```

New data is loaded and scaled in the same manner as the training data.

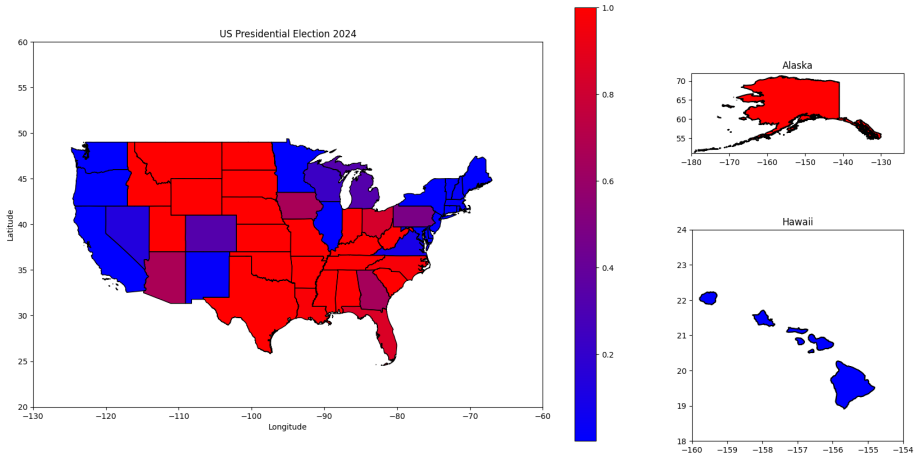
3.9 Make Predictions

```
model.eval()
with torch.no_grad():
    predictions = model(X_new).numpy().flatten().tolist()
    y_pred.extend(predictions)
```

The trained model is used to make predictions on the new data. The model is set to evaluation mode using `model.eval()`, and the predictions are made without computing gradients using `torch.no_grad()`. The predictions are then stored in a list.

4 Plot the Results

Using the Shapefile package and the .shp file available at United States Census Bureau, I plot the just obtained results. As usual, Red stands for Republicans, Blue for Democrats.



5 Conclusion

This process demonstrates the use of a Probit regression model to predict election results based on a variety of macroeconomic indicators. The methodology ensures that the model is trained on historical data and is capable of making informed predictions on new data, potentially aiding in economic forecasting and policy-making decisions.

5.1 Results

After estimating the model, we obtain the estimated coefficients $\hat{\boldsymbol{\beta}}$. The fitted probit model is the following.

$$P(Y = 1|X) = \Phi(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \hat{\beta}_3 X_3 + \hat{\beta}_4 X_4 + \hat{\beta}_5 X_5)$$

The results are not final. Each 1^{st} of the month new data will come up from FRED and I will tempestively re-run the code to come up with the most accurate possible result.

6 Sources

In order to allow for reproducibility I will list here all the sources used to construct the DataSet:

- The data for GDP Growth: FRED
- The data for YoY Inflation: FRED
- The data for House's Price FRED

- The data for Electoral results US History Archive
- The data for Share of Collage Graduates FRED
- The data for Personal Income BEA