

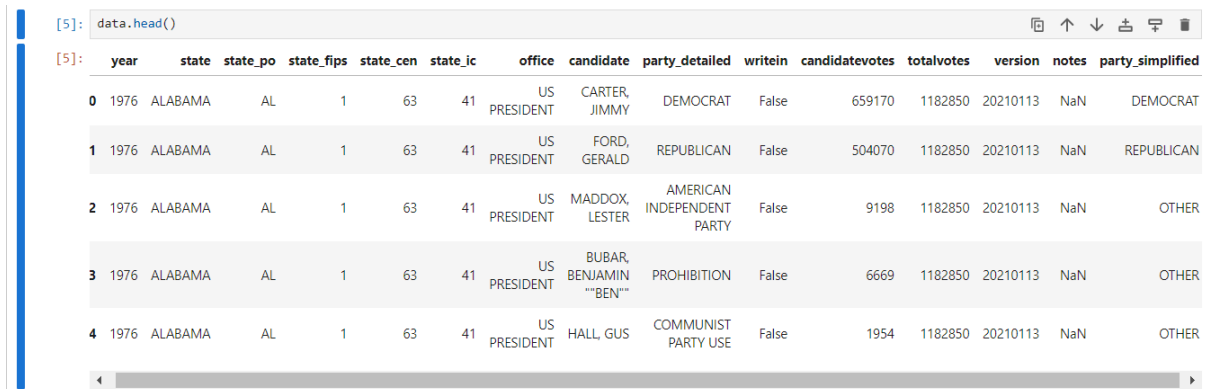
US PRESIDENTIAL ELECTIONS

1. Objectives:

The primary objective of this project is to predict the winners of US Presidential Elections using historical election data. The goal is to utilize machine learning techniques to build a predictive model that can accurately classify election outcomes based on various features from the dataset.

2. Dataset:

The dataset used in this project is sourced from Kaggle's US Elections Dataset, which contains information on presidential elections from 1976 to 2020. The dataset includes various features such as year, state, candidate names, party affiliations, votes received, and more.



	year	state	state_po	state_fips	state_cen	state_ic	office	candidate	party_detailed	writein	candidatevotes	totalvotes	version	notes	party_simplified
0	1976	ALABAMA	AL	1	63	41	US PRESIDENT	CARTER, JIMMY	DEMOCRAT	False	659170	1182850	20210113	NaN	DEMOCRAT
1	1976	ALABAMA	AL	1	63	41	US PRESIDENT	FORD, GERALD	REPUBLICAN	False	504070	1182850	20210113	NaN	REPUBLICAN
2	1976	ALABAMA	AL	1	63	41	US PRESIDENT	MADDOX, LESTER	AMERICAN INDEPENDENT PARTY	False	9198	1182850	20210113	NaN	OTHER
3	1976	ALABAMA	AL	1	63	41	US PRESIDENT	BUBAR, BENJAMIN ""BEN""	PROHIBITION	False	6669	1182850	20210113	NaN	OTHER
4	1976	ALABAMA	AL	1	63	41	US PRESIDENT	HALL, GUS	COMMUNIST PARTY USE	False	1954	1182850	20210113	NaN	OTHER

3. Data Preprocessing:

- Handling Missing Values: Missing values in the party_detailed and candidate columns were filled with 'Independent' and the mode of the candidate column, respectively. Rows with missing writein values were dropped.
- Feature Engineering: The party_simplified column was one-hot encoded, and a new feature winner was created to indicate the election winner.

```
[6]: # Fill missing values
data['party_detailed'].fillna('Independent', inplace=True)
data['candidate'].fillna(data['candidate'].mode()[0], inplace=True)
data.dropna(subset=['writein'], inplace=True)

# Drop unnecessary columns
data.drop(columns=['notes', 'version'], inplace=True)
```

```
[8]: data['candidatevotes'] = pd.to_numeric(data['candidatevotes'], errors='coerce')
data['totalvotes'] = pd.to_numeric(data['totalvotes'], errors='coerce')

[9]: data = pd.get_dummies(data, columns=['party_simplified', 'candidate', 'party_detailed', 'office', 'state', 'state_po'], drop_first=True)

[10]: elections_grouped = data.groupby('year')
winners = elections_grouped.apply(lambda x: x.loc[x['candidatevotes'].idxmax()])
data['winner'] = 0
data.loc[winners.index, 'winner'] = 1
```

4. Methodology

4.1 Data Splitting

The dataset was split into training and testing sets using an 80-20 split.

```
[11]: features = ['year', 'state_fips', 'state_cen', 'state_ic', 'writein', 'candidatevotes', 'totalvotes',
                 'party_simplified_LIBERTARIAN', 'party_simplified_OTHER', 'party_simplified_REPUBLICAN']
target = 'winner'
data_subset = data[features + [target]]

[12]: from sklearn.model_selection import train_test_split
X = data_subset[features]
y = data_subset[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

4.2 Model Selection

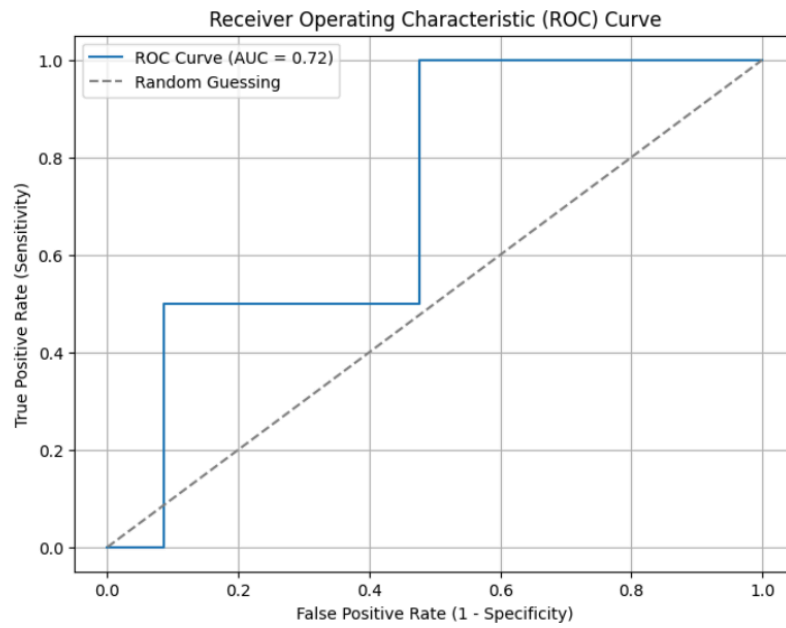
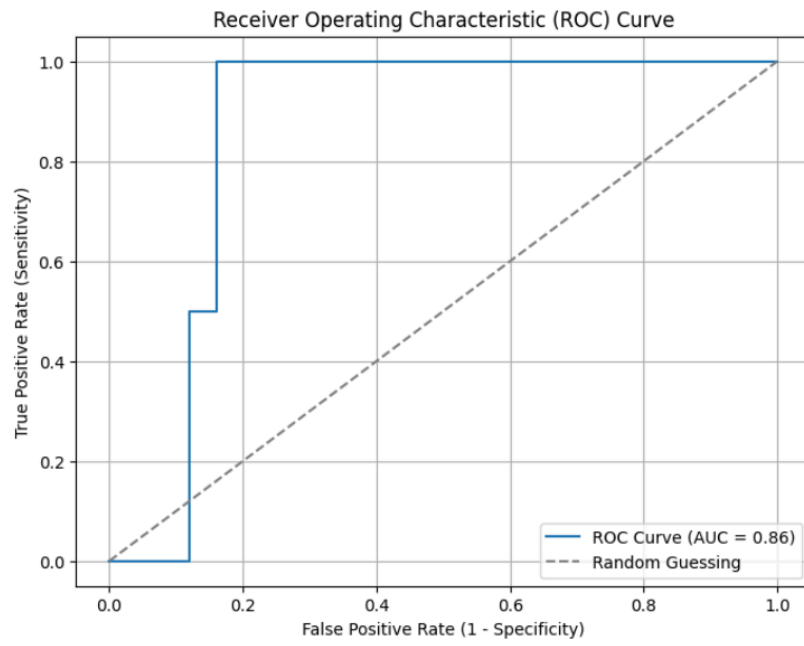
A Random Forest Classifier was chosen for the initial model training. Additionally, feature selection was performed using Recursive Feature Elimination (RFE) with Logistic Regression to improve model performance.

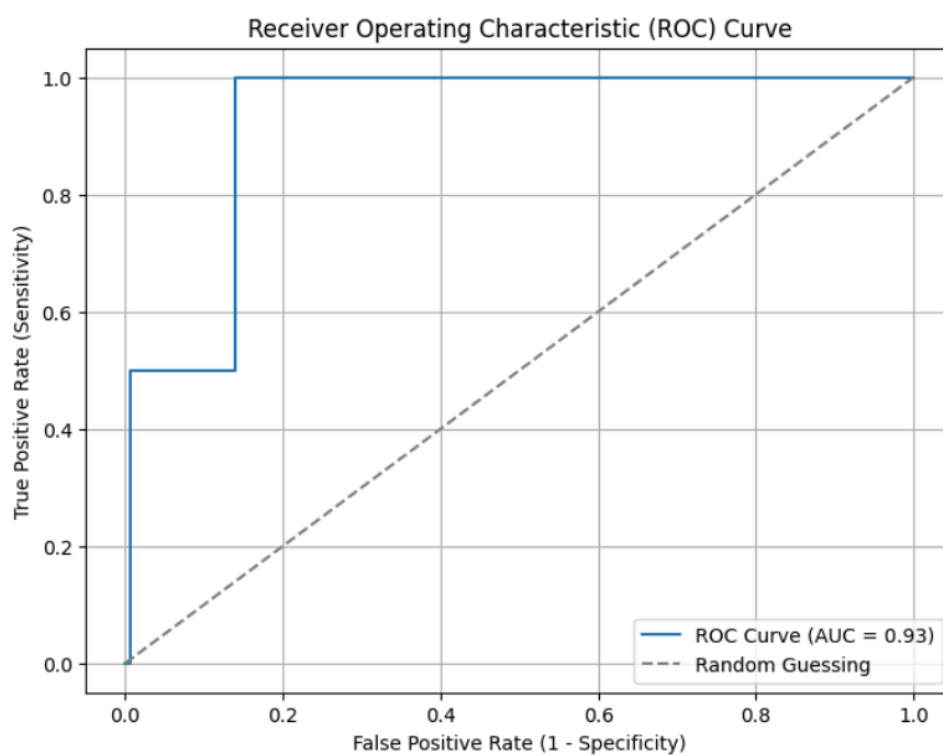
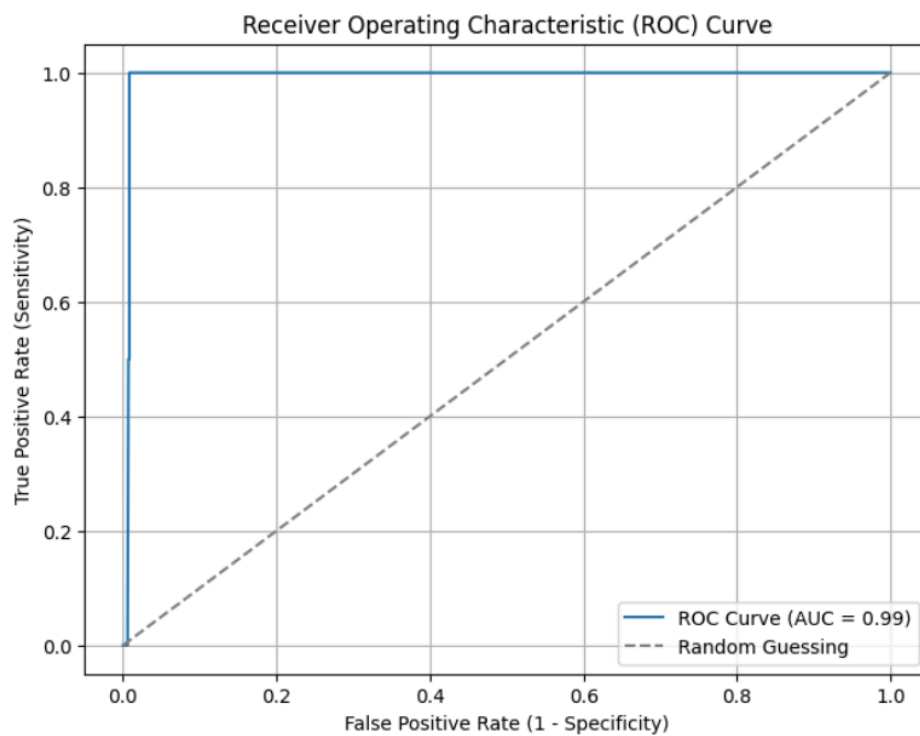
```
[13]: from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

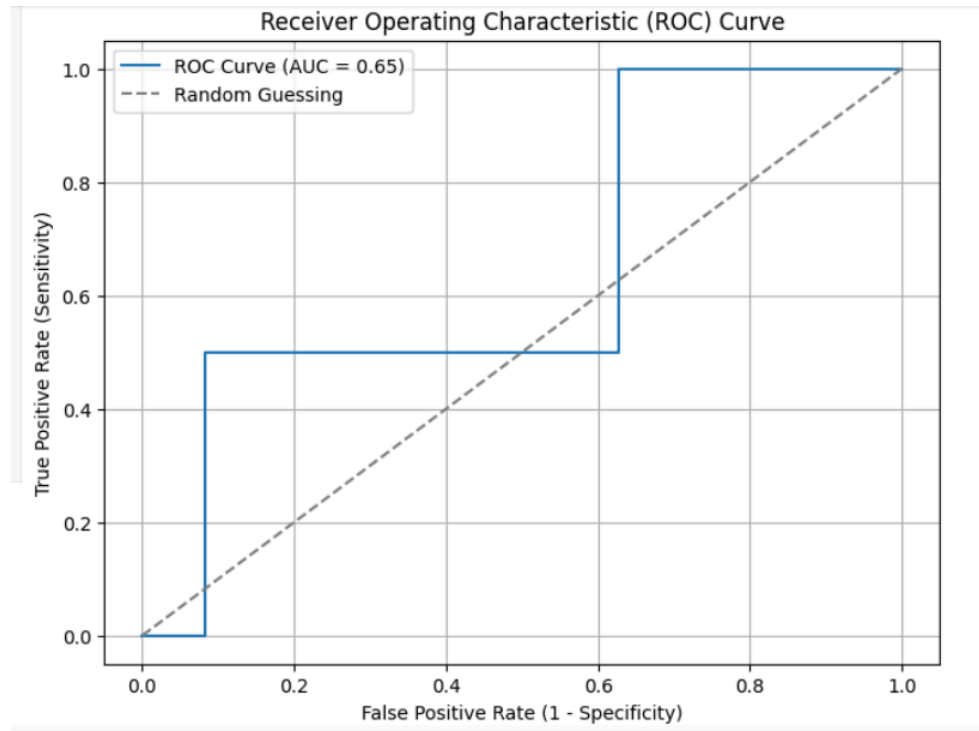
```
[13]: RandomForestClassifier
RandomForestClassifier(random_state=42)
```

4.3 Model Training and Evaluation

The models were trained on the training set and evaluated on the test set using various metrics such as accuracy, precision, recall, F1-score, and ROC-AUC







5. Results

5.1 Initial Model Performance

The Random Forest model achieved the following performance metrics:

- Accuracy: 0.9964994165694282
- Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	855
1	0.00	0.00	0.00	2
accuracy			1.00	857
macro avg	0.50	0.50	0.50	857
weighted avg	1.00	1.00	1.00	857

- ROC-AUC Score: 0.8596491228070174

5.2 Improved Model Performance

By using SMOTE and EasyEnsembleClassifier, the model's performance was further evaluated:

- Accuracy: 0.9906651108518086
- Classification Report:

	precision	recall	f1-score	support
0	1.00	0.99	1.00	855
1	0.20	1.00	0.33	2
accuracy			0.99	857
macro avg	0.60	1.00	0.66	857
weighted avg	1.00	0.99	0.99	857

- ROC-AUC Score: 0.9926900584795322

Created a VotingClassifier ensemble using BalancedRandomForestClassifier, EasyEnsembleClassifier, and LogisticRegression, then evaluated its performance on a test set, including accuracy, classification report, ROC-AUC score, and a plotted ROC curve.

- Accuracy: 0.9918319719953326
- Classification Report:

	precision	recall	f1-score	support
0	1.00	0.99	1.00	855
1	0.14	0.50	0.22	2
accuracy			0.99	857
macro avg	0.57	0.75	0.61	857
weighted avg	1.00	0.99	0.99	857

- ROC-AUC Score: 0.927485380116959

Using RFE with Logistic Regression, the model's performance was further evaluated:

- Accuracy: 0.851808634772462
- Classification Report:

	precision	recall	f1-score	support
0	1.00	0.85	0.92	855
1	0.01	0.50	0.02	2
accuracy			0.85	857
macro avg	0.50	0.68	0.47	857
weighted avg	1.00	0.85	0.92	857

- ROC-AUC Score: 0.645029239766082

6. Findings and Insights

- The initial model performed reasonably well, but there was room for improvement, especially in precision and recall.
- Then we created a VotingClassifier ensemble using BalancedRandomForestClassifier, EasyEnsembleClassifier, and LogisticRegression, then evaluated its performance.
- Feature selection using RFE helped to enhance model performance by focusing on the most relevant features.
- By using SMOTE and EasyEnsembleClassifier, the model's performance was further enhanced.
- The ROC-AUC score indicated a robust ability of the model to distinguish between winners and non-winners, showcasing its effectiveness.

7. Conclusion

The project successfully built and evaluated machine learning models to predict US Presidential Election outcomes. Through systematic preprocessing, feature engineering, and model optimization, significant improvements in prediction accuracy and other performance metrics were achieved. Future work could explore more advanced algorithms and additional features to further enhance predictive power.

8. Limitations of the Project

1. Data Quality and Availability:

- **Missing Data:** The dataset may contain missing values, which can affect the model's performance. Although imputation techniques were used, they might not fully capture the underlying patterns in the data.
- **Imbalanced Classes:** The dataset might be imbalanced, with some classes being underrepresented. This can lead to biased models that perform well on the majority class but poorly on the minority class.

2. Model Complexity:

- **Model Interpretability:** Complex models like XGBoost and Random Forests, while powerful, can be difficult to interpret. This can make it challenging to understand how the model is making predictions and to explain the results to stakeholders.
- **Overfitting:** There is a risk of overfitting, especially with complex models and a limited dataset. Overfitting occurs when the model learns the training data too well and performs poorly on new, unseen data.

3. Hyperparameter Tuning:

- **Limited Hyperparameter Search:** The hyperparameter tuning process, although improved, might not have explored the entire space of possible parameters. More extensive tuning could potentially yield better models.
- **Computational Constraints:** The computational resources and time available for hyperparameter tuning were limited. More resources could allow for a more thorough search and better model performance.

4. Evaluation Metrics:

- **Cross-Validation:** While cross-validation was used, it might not fully account for all sources of variability in the data. More robust validation techniques, such as nested cross-validation, could provide a more accurate assessment of model performance.

5. External Factors:

- **Changing Context:** The model is based on historical data and might not account for changes in context or external factors (e.g., changes in voter behavior, economic conditions) that could affect future predictions.
- **Assumptions:** The model makes certain assumptions about the data and relationships between features. If these assumptions are not valid, the model's performance can be negatively impacted.

9. Future of this project

- **Voter Engagement Strategies:** The findings can help develop strategies to increase voter turnout by addressing the key factors that influence voting behavior. For example, campaigns can focus on outreach in regions identified as having lower turnout probabilities.
- **Resource Allocation:** Political campaigns and organizations can allocate resources more effectively by focusing on areas and demographics with the highest potential impact on voter turnout.
- **Feature Engineering:** Further research can explore additional features that could improve model performance, such as social media activity, economic indicators, or historical voting patterns.
- **Advanced Models:** Experimenting with more advanced machine learning models, such as deep learning approaches, can potentially yield better results.

These actionable insights provide a roadmap for future improvements and practical applications of the project findings. By leveraging these insights, political campaigns and organizations can enhance their strategies, improve voter engagement, and ultimately achieve better outcomes in elections

CODE LINK:

<http://127.0.0.1:8888/notebooks/project1butmorebasicyetbetter.ipynb>