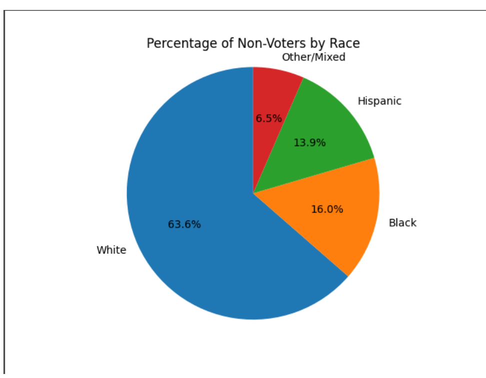
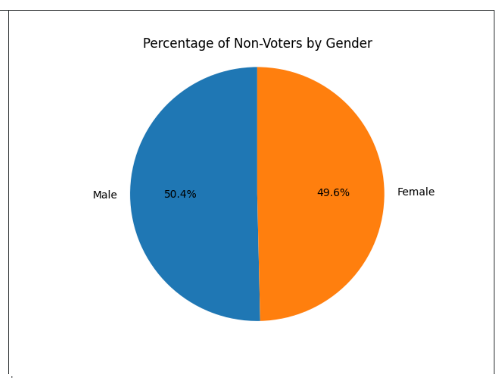
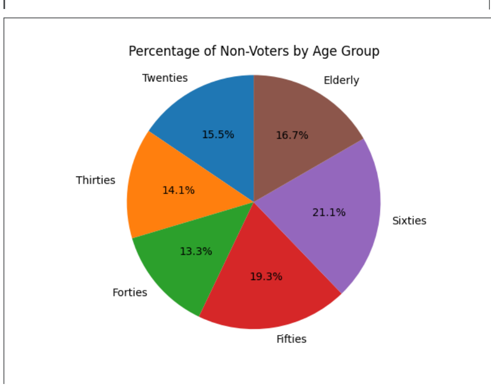
Divya Parmar
May 2, 2021
DATS 6103

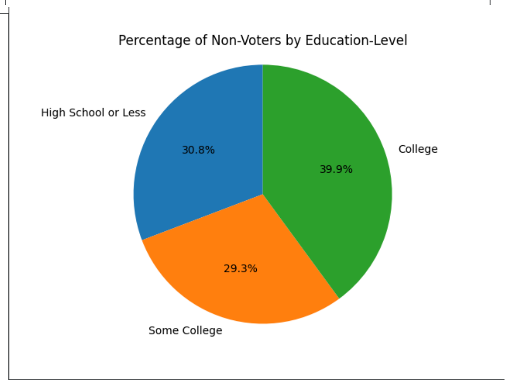
Final Project Individual Write-up

1. Introduction
    a. Using machine learning classification techniques, we predict which candidate a voter plans to vote for based on self-reported survey data.
    b. We chose this dataset because the topic is highly relevant, there was an opportunity to apply EDA and preprocessing to a large number of features, and the dataset was mostly clean and well-documented courtesy of FiveThirtyEight.
    c. The shared work included data cleaning, exploratory data analysis, preprocessing, modeling, model iteration, GUI development, the write-up, and the demo.
2. Description of Individual Work
    a. I found the non-voters dataset, suggested it to my group, and went through the documentation to understand the columns. I re-named all 100+ columns to allow for better readability.
    b. I wrote code to read in the data, label encode the variables, replace invalid feature values with a mean, and run a function to fit a random forest for a pre-specified number of features.
    c. I contributed to the group write-up and PowerPoint by adding the introduction, dataset explanation, and conclusion. I spoke to those sections in our presentation recording.
3. Description of My Portion in Detail
    a. Please see the code on appended at the end of this report.
4. Results









Model 1 Random Forest - Full Model	Model 2 Random Forest - Slim Model	Model 3 Gradient Boosting - Full Model	Model 4 Gradient Boosting - Slim Model
F1-score: 0.98 Accuracy score: 0.97	F1-score: 0.93 Accuracy score: 0.93	F1-score: 0.97 Accuracy score: 0.97	F1-score: 0.50 Accuracy score: 0.51

5. <u>Summary</u>
    a. Using FiveThirtyEight survey data, we decided to predict who voters would vote for president based on their survey answers.
    b. We conducted exploratory data analysis to better understand the group of voters and make sure the classes were balanced.
    c. We preprocessed our data - label encoding, dropping columns and observations.
    d. We fit both random forest and gradient boosting models, and we ran on "full" and "slim" feature sets.
    e. We saw extremely high accuracy and f1 scores, as evidenced in "Results" section.
    f. Random forest did better than gradient boosting, and "full" feature models did better than "slim" feature models.
6. <u>Percent of Code</u>
    a. 0%. No code was copied from the internet for this. I looked up function and syntax, but all EDA/preprocessing/modeling code was written on our own.
7. <u>References</u>

    a. https://scikit-learn.org
    b. https://numpy.org/
    c. https://pandas.pydata.org/
    d. https://pypi.org/project/PyQt5/
    e. https://medium.com/analytics-vidhya/evaluating-a-random-forest-model-9d165595ad56
    f. https://www.datasciencecentral.com/profiles/blogs/decision-tree-vs-random-forest-vs-boosted-trees-explained#:~:text=Like%20random%20forests%2C%20gradient%20boosting,one%20tree%20at%20a%20time
    g. https://morningconsult.com/opinions/to-persuade-or-to-turn-out-voters-is-that-the-question/
    h. https://www.bloomberg.com/graphics/2020-us-election-results/methodology

Appendix – My Individual Code

```
'''

#---------------------------------------------------------------------------
-----------------------------
FiveThirtyEight Non-Voters Dataset
#---------------------------------------------------------------------------
-----------------------------

'''

##### Import packages and data #####

import pandas as pd
import numpy as np
import os
from pathlib import Path
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import plot_roc_curve
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_auc_score

dir = str(Path(os.getcwd()).parents[0])
df = pd.read_csv(dir+'\\'+'nonvoters_data.csv', sep=',', header=0)

# Change directory for graphing purposes
graphing_dir = os.path.join(dir, 'Graphs')
if not os.path.exists(graphing_dir):
    os.mkdir(graphing_dir)
os.chdir(graphing_dir)

##### Exploratory data analysis ##### ---------------------------------------
-----------------------------------

print(df.head)
initial_cols = df.columns
print([x for x in df.columns])

print(df['Q1'].value_counts())
print(df['ppage'].value_counts())
print(df['educ'].value_counts())
print(df['race'].value_counts())
print(df['gender'].value_counts())
print(df['income_cat'].value_counts())
print(df['voter_category'].value_counts())

#### Data Pre-Processing to prepare for modeling ##### ----------------------
----------------------------------------------------
```

```
# Rename columns to descriptive names
df.columns = ['RespId', 'weight',
              'q1_uscitizen',

'q2_important_voting','q2_important_jury','q2_important_following','q2_import
ant_displaying','q2_important_census',

'q2_important_pledge','q2_important_military','q2_important_respect','q2_impo
rtant_god','q2_important_protesting',

'q3_statement_racism1','q3_statement_racism2','q3_statement_feminine',

'q3_statement_msm','q3_statement_politiciansdontcare','q3_statement_besensiti
ve',

'q4_impact_officialsfed','q4_impact_officialsstate','q4_impact_officialslocal
',

'q4_impact_news','q4_impact_wallstreet','q4_impact_lawenforcement',
              'q5_electionmatters',
              'q6_officialsarelikeyou',
              'q7_governmentdesign',

'q8_trust_presidency','q8_trust_congress','q8_trust_supremecourt','q8_trust_c
dc','q8_trust_electedofficials',

'q8_trust_fbicia','q8_trust_newsmedia','q8_trust_police','q8_trust_postalserv
ice',

'q9_politicalsystems_democracy','q9_politicalsystems_experts','q9_politicalsy
stems_strongleader','q9_politicalsystems_army',

'q10_disability','q10_chronic_illness','q10_unemployed','q10_evicted',
              'q11_lostjob','q11_gotcovid','q11_familycovid',
              'q11_coviddeath','q11_worriedmoney','q11_quitjob',
              'q14_view_of_republicans',
              'q15_view_of_democrats',
              'q16_how_easy_vote',

'q17_secure_votingmachines','q17_secure_paperballotsinperson','q17_secure_pap
erballotsmail','q17_secure_electronicvotesonline',

'q18_votingsituations1','q18_votingsituations2','q18_votingsituations3','q18_
votingsituations4','q18_votingsituations5',

'q18_votingsituations6','q18_votingsituations7','q18_votingsituations8','q18_
votingsituations9','q18_votingsituations10',

'q19_get_more_voting1','q19_get_more_voting2','q19_get_more_voting3','q19_get
_more_voting4','q19_get_more_voting5',

'q19_get_more_voting6','q19_get_more_voting7','q19_get_more_voting8','q19_get
_more_voting9','q19_get_more_voting10',
              'q20_currentlyregistered',
              'q21_plan_to_vote',
              'q22_whynotvoting_2020',
```

```
              'q23_which_candidate_supporting',
              'q24_preferred_voting_method',
              'q25_howcloselyfollowing_election',
              'q26_which_voting_category',
              'q27_didyouvotein18','q27_didyouvotein16','q27_didyouvotein14',
              'q27_didyouvotein12','q27_didyouvotein10','q27_didyouvotein08',

'q28_whydidyouvote_past1','q28_whydidyouvote_past2','q28_whydidyouvote_past3'
,'q28_whydidyouvote_past4',

'q28_whydidyouvote_past5','q28_whydidyouvote_past6','q28_whydidyouvote_past7'
,'q28_whydidyouvote_past8',

'q29_whydidyounotvote_past1','q29_whydidyounotvote_past2','q29_whydidyounotvo
te_past3','q29_whydidyounotvote_past4','q29_whydidyounotvote_past5',

'q29_whydidyounotvote_past6','q29_whydidyounotvote_past7','q29_whydidyounotvo
te_past8','q29_whydidyounotvote_past9','q29_whydidyounotvote_past10',
              'q30_partyidentification',
              'q31_republicantype',
              'q32_democratictype',
              'q33_closertowhichparty',
              'ppage', 'educ', 'race', 'gender', 'income_cat',
'voter_category'
              ]

# Drop irrelevant fields (US Citizen, responder ID, observation weight)
# Drop questions that were not asked to all participants (i.e. "why did you
vote" to non-voters, "Republican type" for Democrats)
df.drop(['q1_uscitizen','q22_whynotvoting_2020',

'q28_whydidyouvote_past1','q28_whydidyouvote_past2','q28_whydidyouvote_past3'
,'q28_whydidyouvote_past4',

'q28_whydidyouvote_past5','q28_whydidyouvote_past6','q28_whydidyouvote_past7'
,'q28_whydidyouvote_past8',

'q29_whydidyounotvote_past1','q29_whydidyounotvote_past2','q29_whydidyounotvo
te_past3','q29_whydidyounotvote_past4','q29_whydidyounotvote_past5',

'q29_whydidyounotvote_past6','q29_whydidyounotvote_past7','q29_whydidyounotvo
te_past8','q29_whydidyounotvote_past9','q29_whydidyounotvote_past10',
              'q31_republicantype',
              'q32_democratictype',
              'q33_closertowhichparty',
         'q21_plan_to_vote',
         'q22_whynotvoting_2020',
         'RespId',
         'weight'
         ], axis=1, inplace=True)



# Replace "refused" answers (value of -1) with the demographic average for
each group
# Step 1 - Replace -1 in certain columns with NaN
# Step 2 - Replace NaN with demographic average using groupby
```

```
# Create list of columns that need answer cleaning
# This isn't all the columns (some columns only had values of -1 and 1, which is fine)
replace_neg_one = [

'q2_important_voting','q2_important_jury','q2_important_following','q2_important_displaying','q2_important_census',

'q2_important_pledge','q2_important_military','q2_important_respect','q2_important_god','q2_important_protesting',

'q3_statement_racism1','q3_statement_racism2','q3_statement_feminine',

'q3_statement_msm','q3_statement_politiciansdontcare','q3_statement_besensitive',

'q4_impact_officialsfed','q4_impact_officialsstate','q4_impact_officialslocal',

'q4_impact_news','q4_impact_wallstreet','q4_impact_lawenforcement',
               'q5_electionmatters',
               'q6_officialsarelikeyou',
               'q7_governmentdesign',

'q8_trust_presidency','q8_trust_congress','q8_trust_supremecourt','q8_trust_cdc','q8_trust_electedofficials',

'q8_trust_fbicia','q8_trust_newsmedia','q8_trust_police','q8_trust_postalservice',

'q9_politicalsystems_democracy','q9_politicalsystems_experts','q9_politicalsystems_strongleader','q9_politicalsystems_army',

'q10_disability','q10_chronic_illness','q10_unemployed','q10_evicted',
               'q11_lostjob','q11_gotcovid','q11_familycovid',
               'q11_coviddeath','q11_worriedmoney','q11_quitjob',
               'q14_view_of_republicans',
               'q15_view_of_democrats',
               'q16_how_easy_vote',

'q17_secure_votingmachines','q17_secure_paperballotsinperson','q17_secure_paperballotsmail','q17_secure_electronicvotesonline',

'q18_votingsituations1','q18_votingsituations2','q18_votingsituations3','q18_votingsituations4','q18_votingsituations5',

'q18_votingsituations6','q18_votingsituations7','q18_votingsituations8','q18_votingsituations9','q18_votingsituations10',
               'q20_currentlyregistered',
               'q24_preferred_voting_method',
               'q25_howcloselyfollowing_election',
               'q26_which_voting_category',
               'q27_didyouvotein18','q27_didyouvotein16','q27_didyouvotein14',
               'q27_didyouvotein12','q27_didyouvotein10','q27_didyouvotein08',
               'q30_partyidentification'
               ]
```

```
# Step 1 - Replace -1 or -1.0 values with NaN
# Values might be stored as int or float, so account for both
df[replace_neg_one] = df[replace_neg_one].replace(-1, np.nan)
df[replace_neg_one] = df[replace_neg_one].replace(-1.0, np.nan)

# Step 2 - Replace NaN with demographic mean
for x in replace_neg_one:
    df[x] = df[x].fillna(df.groupby(by=['educ', 'race', 'gender',
'income_cat'])[x].transform('mean'))

# Transform non-numeric categorical variables into numeric for model
processing
le = LabelEncoder()
df['educ'] = le.fit_transform(df['educ'])
df['race'] = le.fit_transform(df['race'])
df['gender'] = le.fit_transform(df['gender'])
df['income_cat'] = le.fit_transform(df['income_cat'])
df['voter_category'] = le.fit_transform(df['voter_category'])

# Identify values of the target variable
print(df['q23_which_candidate_supporting'].value_counts())

# For q23_which_candidate_supporting, value of 1 is Trump and value of 2 is
Biden
# Drop unsure (value of 3) and refused to answer (value of -1) to set up our
two-way classification
df_mod = df[(df['q23_which_candidate_supporting'] == 1) |
(df['q23_which_candidate_supporting'] == 2)]


##### Random Forest Model - Full Model with All Features ##### --------------
------------------------------------------------------------

# Create features dataframe that doesn't contain the target variable
X = df_mod.drop(['q23_which_candidate_supporting'], axis=1)
# Create target variable
y = df_mod['q23_which_candidate_supporting']

# Split data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=419)

# Fit model on train data
clf = RandomForestClassifier(n_estimators=500)
clf.fit(X_train, y_train)
# Predict on test data
# Categorical predictions are for accuracy and probabilities are for ROC
score
y_pred = clf.predict(X_test)
y_pred_probs = clf.predict_proba(X_test)

# Get accuracy and ROC values
roc_auc_full = roc_auc_score(y_test, y_pred_probs[:, 1])
accuracy_full = accuracy_score(y_test, y_pred)
print(f'The full model AUC is {roc_auc_full} and the accuracy is
{accuracy_full}.')
```

```
# Plot ROC curve
# More area under the curve indicates the model has skill in finding true
positives and avoiding false positives
plot_roc_curve(clf, X_test, y_test)
plt.savefig('roc_curve_full_model.png', dpi=300, bbox_inches='tight')
plt.show()

# Get feature importances and plot them
importances = clf.feature_importances_
feat_imp = pd.Series(importances, X_train.columns)
feat_imp.sort_values(ascending=False, inplace=True)
feat_imp.plot(x='Features', y='Importance', kind='bar', figsize=(16, 9),
rot=90, fontsize=15)
plt.tight_layout()
plt.savefig('feature_importances_full_model.png', dpi=300,
bbox_inches='tight')
plt.show()

##### Feature Importance Analysis ##### ------------------------------------
-------------------------------------

# Get top 20 features
top20 = feat_imp.index[0:20]

# Plot correlation matrix of top 20 features against the target variable (for
all records)
df20 = df[top20]
df20['y'] = df[y.name]

plt.figure(figsize=(16,16))
plt.tight_layout()
sns.set(font_scale=1)
corr_heatmap = sns.heatmap(df20.corr(), vmin=-1, vmax=1, annot=True,
cbar=False)
corr_heatmap.set_title('Correlation of Top 20 Features and Target Variable')
corr_heatmap.set_xticklabels(labels=df20.columns, rotation=30, fontsize=9,
ha='right')
plt.savefig('heatmap_top20_features.png', dpi=300, bbox_inches='tight')
plt.show()

##### Random Forest Model - Slim model without the top features ##### -------
-------------------------------------------------------------------

# Run another model without top features such as party identification and
trust of presidency
# These variables are very highly correlated with view of Trump, GOP, Dems,
etc.
X_slim = df_mod.drop(['q23_which_candidate_supporting',
'q30_partyidentification','q8_trust_presidency',
                 'q14_view_of_republicans', 'q15_view_of_democrats'], axis=1)

# Train test split for this new model
X_slim_train, X_slim_test, y_slim_train, y_slim_test =
train_test_split(X_slim, y, test_size=0.25, random_state=125)

# Fit model
```

```
clf2 = RandomForestClassifier(n_estimators=500)
clf2.fit(X_slim_train, y_slim_train)
# Predict
y_slim_pred = clf2.predict(X_slim_test)
y_slim_pred_probs = clf2.predict_proba(X_slim_test)

# Get accuracy and ROC values
roc_auc_slim = roc_auc_score(y_slim_test, y_slim_pred_probs[:, 1])
accuracy_slim = accuracy_score(y_slim_test, y_slim_pred)
print(f'The slim model AUC is {roc_auc_slim} and the accuracy is
{accuracy_slim}.')

# Plot ROC curve
plot_roc_curve(clf2, X_slim_test, y_slim_test)
plt.savefig('roc_curve_slim_model.png', dpi=300, bbox_inches='tight')
plt.show()

# Get feature importances
importances2 = clf2.feature_importances_
feat_imp2 = pd.Series(importances2, X_slim_train.columns)
feat_imp2.sort_values(ascending=False, inplace=True)
feat_imp2.plot(x='Features', y='Importance', kind='bar', figsize=(16, 9),
rot=90, fontsize=15)
plt.tight_layout()
plt.savefig('feature_importances_slim_model.png', dpi=300,
bbox_inches='tight')
plt.show()


##### IGNORE CODE BELOW ##### ---------------------------------------------------
---------------------------

'''

print(df['q21_plan_to_vote'].value_counts())
print(df['q30_partyidentification'].value_counts())


df.drop(['q1_uscitizen','q20_currentlyregistered','q22_whynotvoting_2020',
              'q23_which_candidate_supporting','q26_which_voting_category',
              'q27_didyouvotein18','q27_didyouvotein16','q27_didyouvotein14',
              'q27_didyouvotein12','q27_didyouvotein10','q27_didyouvotein08',

'q28_whydidyouvote_past1','q28_whydidyouvote_past2','q28_whydidyouvote_past3'
,'q28_whydidyouvote_past4',

'q28_whydidyouvote_past5','q28_whydidyouvote_past6','q28_whydidyouvote_past7'
,'q28_whydidyouvote_past8',

'q29_whydidyounotvote_past1','q29_whydidyounotvote_past2','q29_whydidyounotvo
te_past3','q29_whydidyounotvote_past4','q29_whydidyounotvote_past5',

'q29_whydidyounotvote_past6','q29_whydidyounotvote_past7','q29_whydidyounotvo
te_past8','q29_whydidyounotvote_past9','q29_whydidyounotvote_past10',
              'q31_republicantype',
              'q32_democratictype',
              'q33_closertowhichparty',
```

```
        'voter_category',
        'RespId',
        'weight'
        ], axis=1, inplace=True)


# Fit the random forest
# Get feature
rf = RandomForestClassifier()
cv = cross_validate(rf, X, y, cv=10)
print(cv)

# Get feature importances
rf2 = RandomForestClassifier()
rf2.fit(X=X, y=y, sample_weight=None)
feat_imp = list(zip(rf2.feature_importances_, X.columns))
print(sorted(feat_imp, reverse=True))

# Try feature selection with SelectFromModel
select = SelectFromModel(RandomForestClassifier(n_estimators=20))
select.fit(X_train, y_train)
# Select.get_support returns True or False for each feature
# Take only the true values for features and look at our accuracy
print(select.get_support())
feature_inclusion_array = select.get_support()
print(X_train.columns[feature_inclusion_array])
inclusion_cols = X_train.columns[feature_inclusion_array]
X_train_skinny = X_train[inclusion_cols]

rf3 = RandomForestClassifier()
cv_skinny = cross_validate(rf3, X_train_skinny, y_train, cv=10)
print(cv_skinny)

plt.tight_layout()
#plt.subplots_adjust(top = 3, bottom = 2, right = 3, left = 2,
#            hspace = 0.1, wspace = 0.1)
#corr_heatmap.xaxis.labelpad = 0
#corr_heatmap.title.labelpad = 0

'''


# Write a function that takes
# X, y, percent test in train-test, number of features in model
# Return model accuracy metrics, confusion matrix, feature importance, roc
curve

def rf_model_visualize(df: pd.DataFrame, num_features: int, test_percent:
float):

    '''

    :param df: Dataframe of all observations (train and test) to build model.
    :param num_features: The number of features to include in the model (all
variables except target).
    :param test_percent: Percent of data to use in test (i.e. 0.3 means 70%
train, 30% test).
```

```
    :return: accuracy_score_value: The accuracy of the RF model with the
parameters passed above. (TP + FN)/ (TP + FP + TN + FN)
    :return: conf: Confusion matrix of RF model. This classifies the true
positives, false positives, true negatives, false negatives.
    :return: auc_graph: Graph of AUC (area under curve) of the RF model.
    :return: auc_score_value: AUC (area under curve) score. Random guessing
is 0.5, and closer to 1 means smarter model.
    :return: feature_importance_plot: Importance of the num_features chosen.
Higher importance means it greater reduces entropy in classification.

    '''

    # Create empty variables to return if the user passes in invalid
parameters
    auc_null = np.nan
    conf_null = np.zeros((2,2), dtype=int)
    auc__null_graph = plt.plot()
    auc_score_null = np.nan
    feature_importance_plot_null = plt.plot()

    # There are only 92 features available
    if (num_features < 1) or (num_features > 92):
        return auc_null, conf_null, auc__null_graph, auc_score_null,
feature_importance_plot_null
    # We cannot test on 0 or 100 percent of our data
    if (test_percent < 0.01) or (test_percent > 0.99):
        return auc_null, conf_null, auc__null_graph, auc_score_null,
feature_importance_plot_null

    # Go through modeling steps in this function
    # Start with getting X, y, and train-test split
    Xpre = df.drop(columns=['q23_which_candidate_supporting'], axis=1)
    ypre = df['q23_which_candidate_supporting']

    X_pre_train, X_pre_test, y_pre_train, y_pre_test = train_test_split(Xpre,
ypre, test_size=test_percent, random_state=1918)

    # Fit the model
    rf_pre = RandomForestClassifier()
    rf_pre.fit(X_pre_train, y_pre_train)

    # Get the most important features
    importances = rf_pre.feature_importances_
    feat_imp = pd.Series(importances, X_pre_train.columns)
    feat_imp.sort_values(ascending=False, inplace=True)
    features_to_keep = feat_imp.index[0:num_features]

    # Re-fit with the slimmed down list
    X = Xpre[features_to_keep]
    y = ypre

    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=test_percent, random_state=1918)

    # Fit the model
    rf = RandomForestClassifier()
```

```
    rf.fit(X_train, y_train)
    y_pred = rf.predict(X_test)
    y_pred_proba = rf.predict_proba(X_test)

    # Output: accuracy metrics
    accuracy_score_value = accuracy_score(y_test, y_pred)

    # Output: confusion matrix
    conf = confusion_matrix(y_test, y_pred)

    # Output: ROC Curve
    auc_graph = plot_roc_curve(rf, X_test, y_test)

    # Output: ROC score
    auc_score_value = roc_auc_score(y_test, y_pred_proba[:, 1])

    # Output Feature importance
    imp_final = rf.feature_importances_
    feat_imp_final = pd.Series(imp_final, X_train.columns)
    feat_imp_final.sort_values(ascending=False, inplace=True)
    feature_importance_plot = plt.bar(x=feat_imp_final.index,
height=feat_imp_final.values)

    return accuracy_score_value, conf, auc_graph, auc_score_value,
feature_importance_plot


accuracy_score_value, conf, auc_graph, auc_score_value,
feature_importance_plot = rf_model_visualize(df=df,

num_features=25,

test_percent=0.25)
```