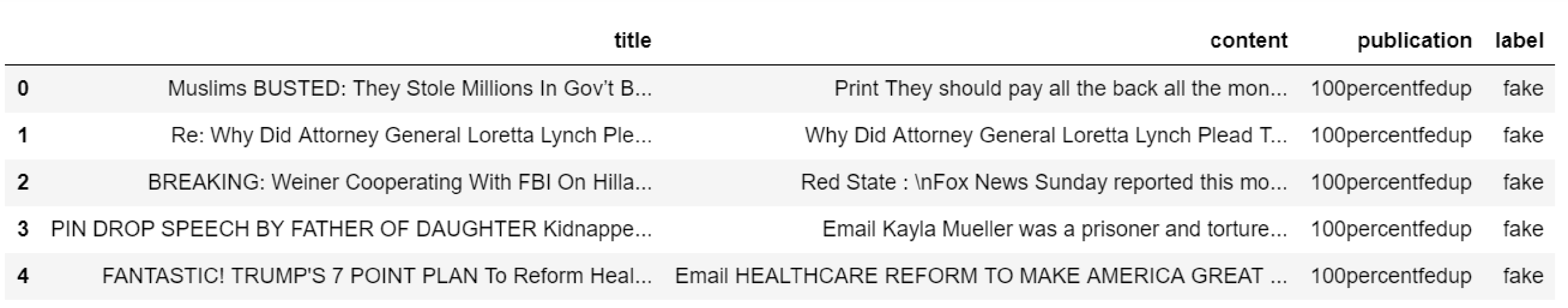
DETECTING FAKE NEWS

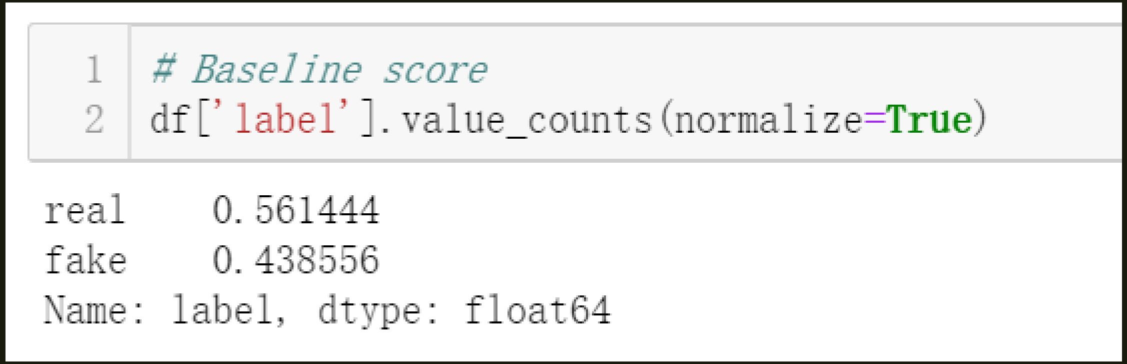
Angie Sheng
30/08/2019

Data Set (Nov, 2016)



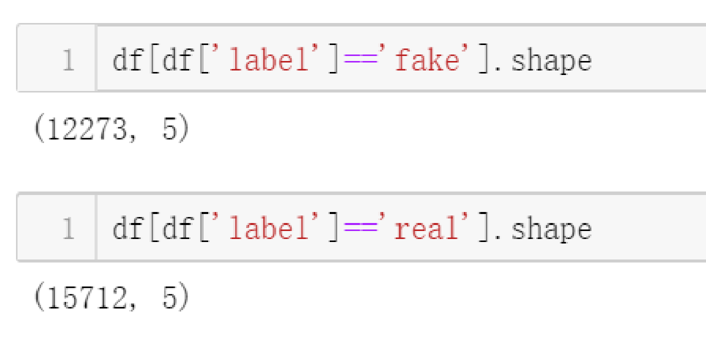
	title	content	publication	label
0	Muslims BUSTED: They Stole Millions In Gov't B...	Print They should pay all the back all the mon...	100percentfedup	fake
1	Re: Why Did Attorney General Loretta Lynch Ple...	Why Did Attorney General Loretta Lynch Plead T...	100percentfedup	fake
2	BREAKING: Weiner Cooperating With FBI On Hilla...	Red State : \nFox News Sunday reported this mo...	100percentfedup	fake
3	PIN DROP SPEECH BY FATHER OF DAUGHTER Kidnappe...	Email Kayla Mueller was a prisoner and torture...	100percentfedup	fake
4	FANTASTIC! TRUMP'S 7 POINT PLAN To Reform Heal...	Email HEALTHCARE REFORM TO MAKE AMERICA GREAT ...	100percentfedup	fake

Baseline Score



```
# Baseline score
df['label'].value_counts(normalize=True)
```

```
real     0.561444
fake     0.438556
Name: label, dtype: float64
```



```
df[df['label']=='fake'].shape
```

```
(12273, 5)
```

```
df[df['label']=='real'].shape
```

```
(15712, 5)
```

Modelling

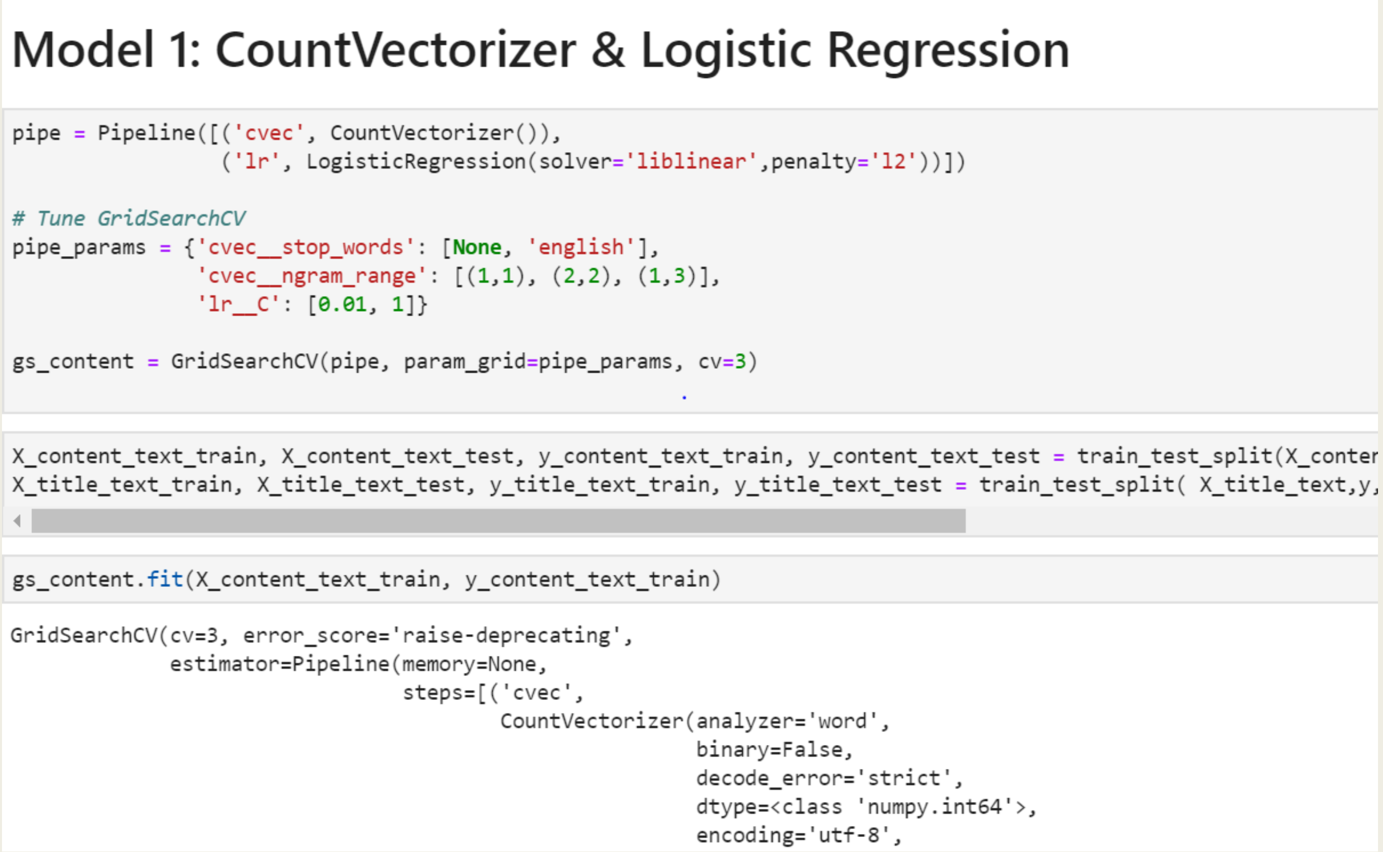
Pipeline & Grid Search:

TFIDF / Count Vectorizer

+

Logistic Regression / Random Forrest / Decision Tree...

My approach is to throw all pipeline and Grid Search thing to Google Cloud



Model 1: CountVectorizer & Logistic Regression

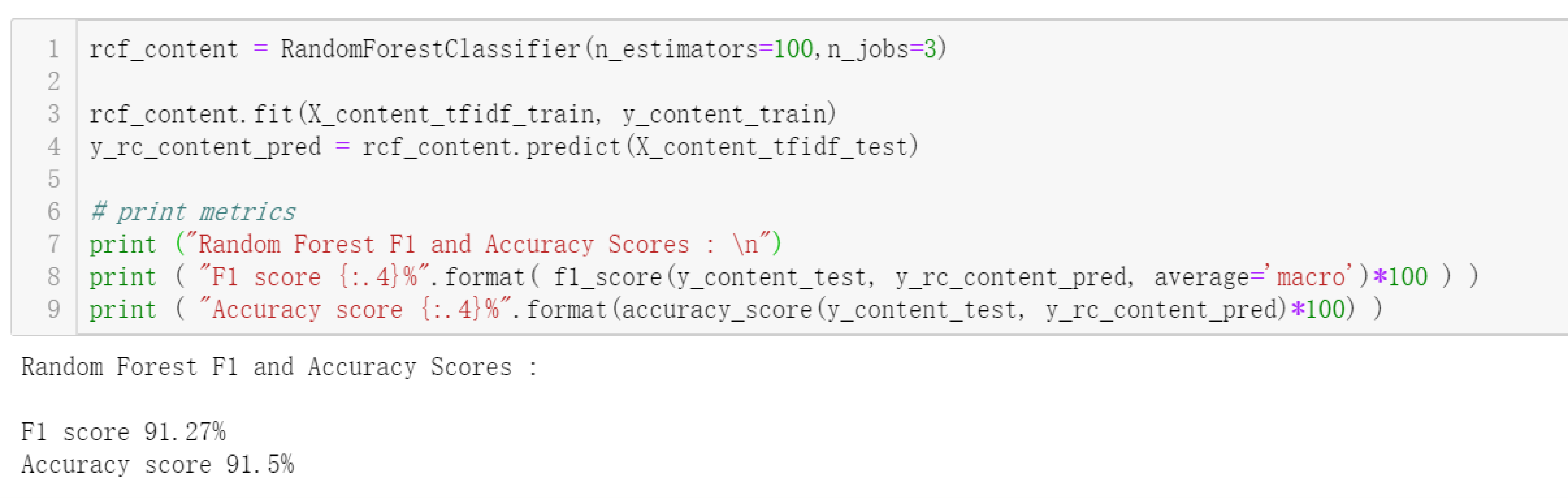
```
pipe = Pipeline([('cvec', CountVectorizer()),
                 ('lr', LogisticRegression(solver='liblinear',penalty='l2'))])

# Tune GridSearchCV
pipe_params = {'cvec__stop_words': [None, 'english'],
               'cvec__ngram_range': [(1,1), (2,2), (1,3)],
               'lr__C': [0.01, 1]}

gs_content = GridSearchCV(pipe, param_grid=pipe_params, cv=3)
```

```
X_content_text_train, X_content_text_test, y_content_text_train, y_content_text_test = train_test_split(X_conter
X_title_text_train, X_title_text_test, y_title_text_train, y_title_text_test = train_test_split( X_title_text,y,
```

```
gs_content.fit(X_content_text_train, y_content_text_train)
```

```
GridSearchCV(cv=3, error_score='raise-deprecating',
             estimator=Pipeline(memory=None,
                                steps=[('cvec',
                                        CountVectorizer(analyzer='word',
                                                        binary=False,
                                                        decode_error='strict',
                                                        dtype=<class 'numpy.int64'>,
                                                        encoding='utf-8',
```

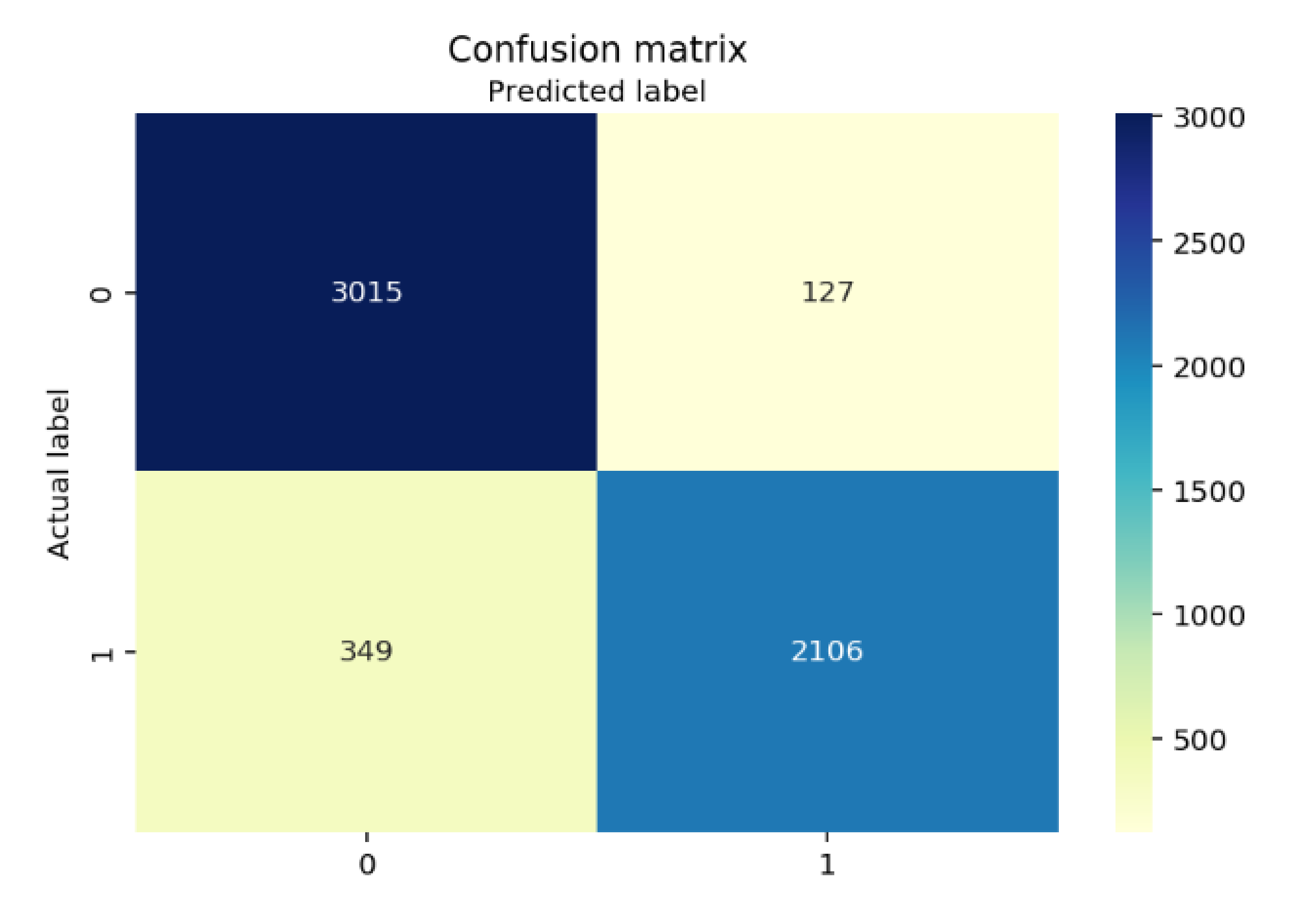
Best Model



```
rcf_content = RandomForestClassifier(n_estimators=100,n_jobs=3)

rcf_content.fit(X_content_tfidf_train, y_content_train)
y_rc_content_pred = rcf_content.predict(X_content_tfidf_test)

# print metrics
print ("Random Forest F1 and Accuracy Scores : \n")
print ( "F1 score {:.4}%".format( f1_score(y_content_test, y_rc_content_pred, average='macro')*100 ) )
print ( "Accuracy score {:.4}%".format(accuracy_score(y_content_test, y_rc_content_pred)*100) )
```

```
Random Forest F1 and Accuracy Scores :

F1 score 91.27%
Accuracy score 91.5%
```

Confusion Matrix



Accuracy:
For all predictions, 91.5 % predicted correctly.

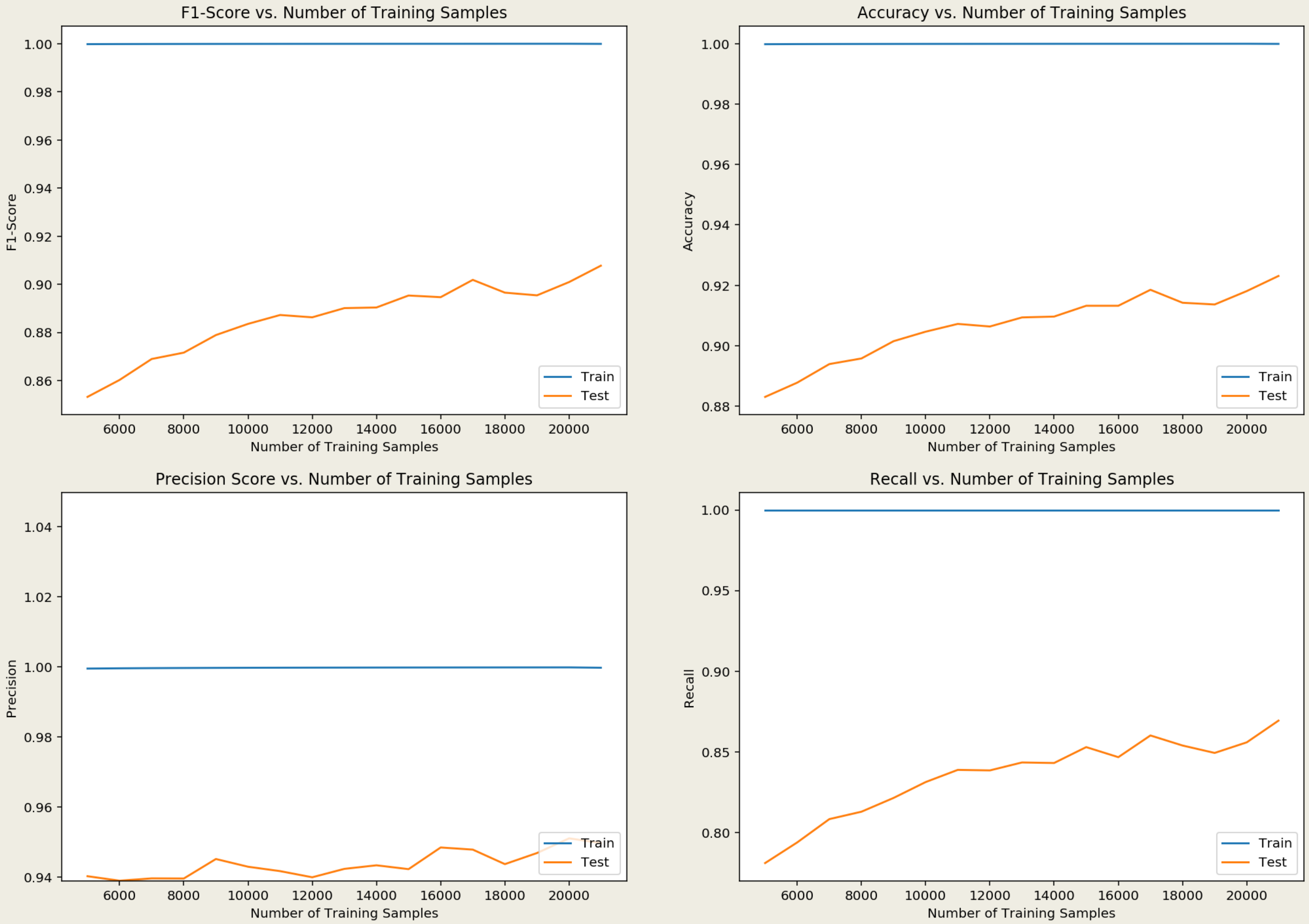
Precision(Positive Predictive Rate):
Among all fake news predictions, 94.31% predicted correctly.

Recall (True Positive Rate):
Among all fake news content, 85.78% are being picked up

Specificity(True Negative Rate):
Among all real news, 95.96 % predicted correctly

Misclassification Rate:
For all predictions, 7.32% predicted incorrectly.

Best Model



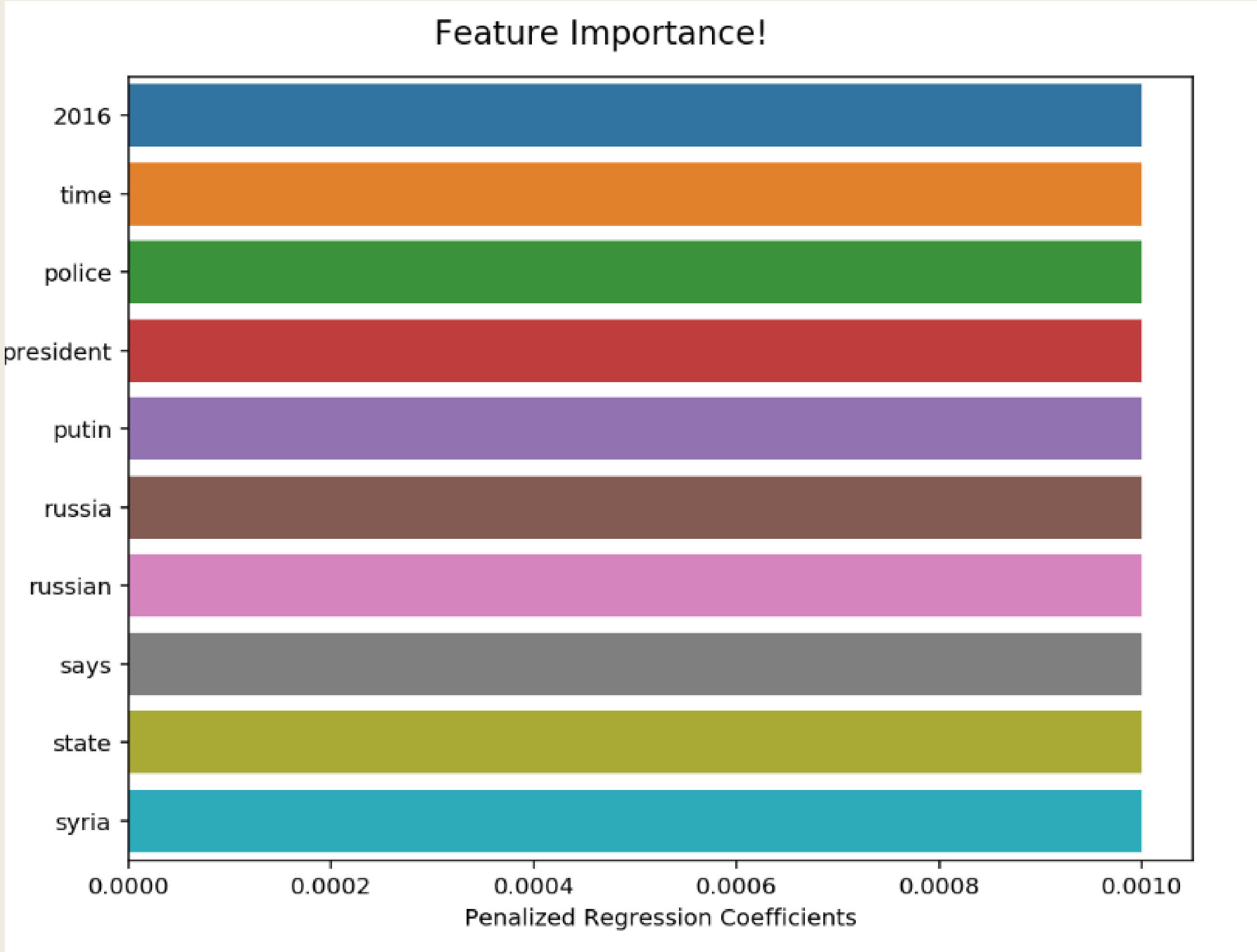
F1-Score vs. Number of Training Samples

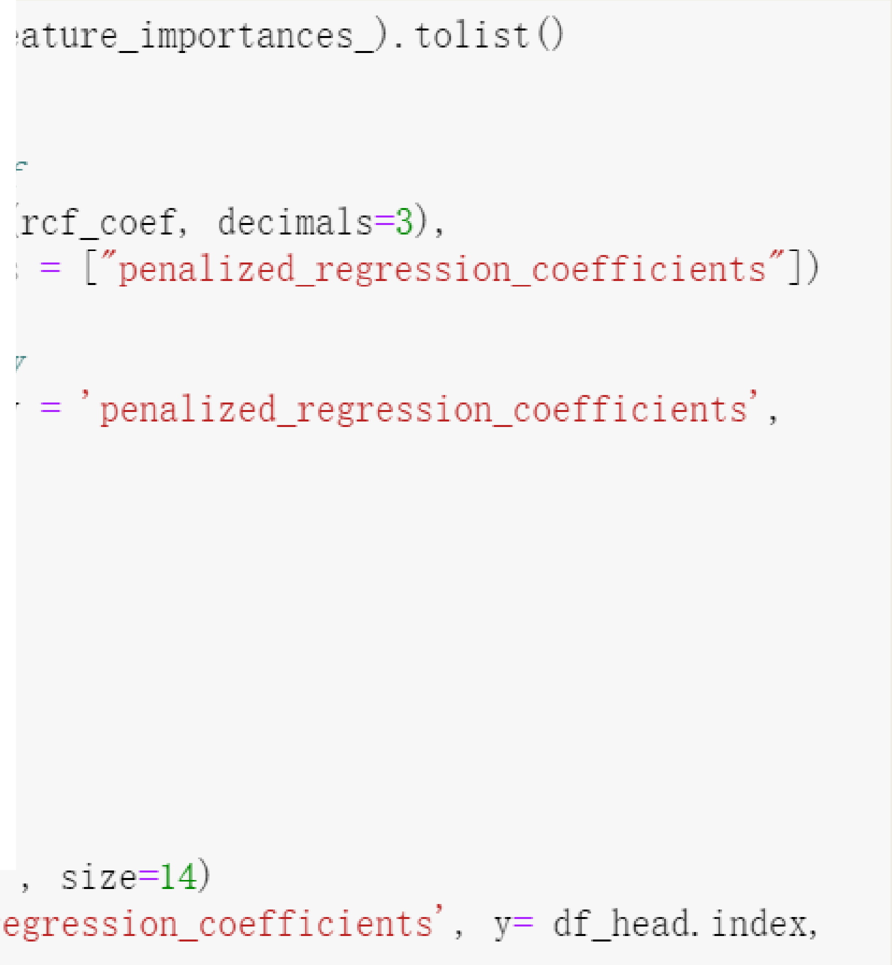
Accuracy vs. Number of Training Samples

Precision Score vs. Number of Training Samples

Recall vs. Number of Training Samples

Feature Importance





```
eature_importances_).tolist()

rcf_coef, decimals=3),
= ["penalized_regression_coefficients"])

= 'penalized_regression_coefficients',

fig.suptitle('Feature Importance!', size=14)
ax = sns.barplot(x = 'penalized_regression_coefficients', y= df_head.index,
data=df_head)
```