In [4]:

```
#data wrangling and visualization packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
import statsmodels.api as sm
import scipy.stats as stats

#feature engineering packages
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder, StandardS

#feature selection packages
from feature_engine.selection import DropDuplicateFeatures
from feature_engine.selection import DropConstantFeatures

#modeling packages
from sklearn.model_selection import train_test_split
from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression, Logis
from sklearn.pipeline import Pipeline
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier

#modeling evaluation packages
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.model_selection import cross_val_score
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import plot_roc_curve, roc_curve, auc
from sklearn.metrics import get_scorer

#optimization packages
from sklearn.model_selection import GridSearchCV
```

0.1 Global Function

In [5]:

```
#function to look at plots and stats of column with or wit
def model_eval(model, X_train, y_train, X_test, y_test,
               prev_model=None, prev_X_train=None, prev_y_
               prev_X_test=None, prev_y_test=None):
    '''
    This function takes in a fit model and provides classi
    metrics on that model. Optionally, a previous model ca
    improvement metrics between the current model and the
    Keyword Arguments:
    - model: A fit model
    - X_train, y_train, X_test, y_test: Training and testi
    the "model" stated above was fit on.
    - prev_model: Another fit model
    - prev_X_train, prev_y_train, prev_X_test, prev_y_test
    testing dataframes which the previous model was fit on
    '''
    #current model predictions on testing dataframe
    y_hat_test = model.predict(X_test)
    #get scores of current model on testing dataframe
    recall_model = get_scorer('recall')(model, X_test, y_t
    f1_model = get_scorer('f1')(model, X_test, y_test).rou
    accuracy_model = get_scorer('accuracy')(model, X_test,
    auc_model = get_scorer('roc_auc')(model, X_test, y_tes

    recall_model_train = get_scorer('recall')(model, X_tra
    #if statement to check for availability of a previous
    #and testing dataframes
    if prev_model != None:
        if prev_X_train is None:
            #if prevoius model has a different training an

            #get previous model predictions and scores
            y_hat_test_prev = prev_model.predict(X_test)

            recall_prev = get_scorer('recall')(prev_model,
            f1_prev = get_scorer('f1')(prev_model, X_test,
            accuracy_prev = get_scorer('accuracy')(prev_mo
            auc_prev = get_scorer('roc_auc')(prev_model, X

            print('MODEL EVAL VS PREVIOUS (TEST)')
            print('=====================================

            #create dataframe comparing current and previo
            df = pd.DataFrame(index=['Recall','F1','Accura
                              columns=['Previous Model','C

            df.loc['Recall','Current Model'] = recall_mode
            df.loc['Recall','Previous Model'] = recall_pre
            df.loc['F1','Current Model'] = f1_model
            df.loc['F1','Previous Model'] = f1_prev
            df.loc['Accuracy','Current Model'] = accuracy_
            df.loc['Accuracy','Previous Model'] = accuracy
            df.loc['AUC','Current Model'] = auc_model
            df.loc['AUC','Previous Model'] = auc_prev
            df.loc['Recall','Delta'] = (recall_model - rec
            df.loc['F1','Delta'] = (f1_model - f1_prev).ro
            df.loc['Accuracy','Delta'] = (accuracy_model -
            df.loc['AUC','Delta'] = (auc_model - auc_prev)

```

Contents


```
60            display(df)
61            print('\n')
62
63            #display current model scores, reports and gra
64            print(f"CURRENT MODEL: {'Overfit (Recall)' if
65            print('======================================
66            print('\n')
67
68            print(f'Recall on Training: {recall_model_trai
69            print(f'Recall on Test: {recall_model}')
70            print('\n')
71
72            print('Classification Reports----------------
73            print(classification_report(y_test, y_hat_test
74
75            print('Test Graphs---------------------------
76            fig, ax = plt.subplots(ncols=2, figsize=(15,8)
77
78            plot_confusion_matrix(model, X_test, y_test, c
79                                  normalize='true',
80                                  display_labels=["NO INJU
81                                  ax=ax[0]);
82
83            plot_roc_curve(model, X_test, y_test, ax=ax[1]
84            plt.show()
85
86            #display previous model graphs
87            print("PREVIOUS MODEL")
88            print('======================================
89            fig, ax = plt.subplots(ncols=2, figsize=(15,8)
90
91            plot_confusion_matrix(prev_model, X_test, y_te
92                                  normalize='true',
93                                  display_labels=['NO INJU
94                                  ax=ax[0]);
95
96            plot_roc_curve(prev_model, X_test, y_test, ax=
97            plt.show()
98
99            plt.tight_layout()
100       else:
101           #if previous model has the same dataframes as
102           y_hat_test_prev = prev_model.predict(prev_X_te
103
104           recall_prev = get_scorer('recall')(prev_model,
105                                              prev_y_test
106           f1_prev = get_scorer('f1')(prev_model, prev_X_
107                                      prev_y_test).round(
108           accuracy_prev = get_scorer('accuracy')(prev_mo
109                                                  prev_y_
110           auc_prev = get_scorer('roc_auc')(prev_model, p
111                                            prev_y_test).
112
113           print('MODEL EVAL VS PREVIOUS (TEST)')
114           print('======================================
115
116           df = pd.DataFrame(index=['Recall','F1','Accura
117                             columns=['Previous Model','C
118
119           df.loc['Recall','Current Model'] = recall_mode
120           df.loc['Recall','Previous Model'] = recall_pre
```

Contents


```
121             df.loc['F1','Current Model'] = f1_model
122             df.loc['F1','Previous Model'] = f1_prev
123             df.loc['Accuracy','Current Model'] = accuracy_
124             df.loc['Accuracy','Previous Model'] = accuracy
125             df.loc['AUC','Current Model'] = auc_model
126             df.loc['AUC','Previous Model'] = auc_prev
127             df.loc['Recall','Delta'] = (recall_model - rec
128             df.loc['F1','Delta'] = (f1_model - f1_prev).ro
129             df.loc['Accuracy','Delta'] = (accuracy_model -
130             df.loc['AUC','Delta'] = (auc_model - auc_prev)
131 
132             display(df)
133             print('\n')
134 
135 
136             print(f"CURRENT MODEL: {'Overfit (Recall)' if
137             print('=======================================
138             print('\n')
139 
140             print(f'Recall on Training: {recall_model_trai
141             print(f'Recall on Test: {recall_model}')
142             print('\n')
143 
144             print('Classification Reports----------------
145             print(classification_report(y_test, y_hat_test
146 
147             print('Test Graphs---------------------------
148             fig, ax = plt.subplots(ncols=2, figsize=(15,8)
149 
150             plot_confusion_matrix(model, X_test, y_test, c
151                                   normalize='true',
152                                   display_labels=['NO INJU
153                                   ax=ax[0]);
154 
155             plot_roc_curve(model, X_test, y_test, ax=ax[1]
156             plt.show()
157 
158             print("PREVIOUS MODEL")
159             print('=======================================
160             fig, ax = plt.subplots(ncols=2, figsize=(15,8)
161 
162             plot_confusion_matrix(prev_model, prev_X_test,
163                                   normalize='true',
164                                   display_labels=['NO INJU
165                                   ax=ax[0]);
166 
167             plot_roc_curve(prev_model, prev_X_test, prev_y
168                            ax=ax[1]).ax_.plot([0,1],[0,1])
169             plt.show()
170 
171             plt.tight_layout()
172     else:
173         #if there is no previous model, get current model
174         print(f"CURRENT MODEL: {'Overfit (Recall)' if reca
175         print('=======================================
176         print('\n')
177         print('Classification Reports----------------
178         print(classification_report(y_test, y_hat_test))
179 
180         print('Test Graphs---------------------------
181         fig, ax = plt.subplots(ncols=2, figsize=(15,8))
```

```
182
183         plot_confusion_matrix(model, X_test, y_test, cmap=
184                                       normalize='true',
185                               display_labels=['NO INJURY',
186                               ax=ax[0]);
187
188         plot_roc_curve(model, X_test, y_test, ax=ax[1]).ax
189         plt.show()
190
191
192     return
```

In [50]:

```
1  def get_coefficients(model,X_train):
2      '''
3      Function to get coefficients of logistic regression mo
4      '''
5
6
7      coeffs = pd.Series(model.coef_.flatten(), index=X_trai
8      coeffs['intercept'] = model.intercept_[0]
9      return coeffs
```

In [6]:

```
1  X_train_dt = pd.read_csv("crash_train.csv")
2  X_test_dt = pd.read_csv("crash_test.csv")
```

In [7]:

```
1  X_train_lr_sel = X_train_dt.copy()
2  X_test_lr_sel = X_test_dt.copy()
```

In [8]:

```
1  y_train = pd.read_csv("crash_y_train.csv")
2  y_test = pd.read_csv("crash_y_test.csv")
3
```

In [9]:

```
1  y_train.drop("Unnamed: 0", axis=1, inplace=True)
2  y_test.drop("Unnamed: 0", axis=1, inplace=True)
```

In [10]:

```
1  X_test_dt.drop("Unnamed: 0", axis=1, inplace=True)
2  X_train_dt.drop("Unnamed: 0", axis=1, inplace=True)
```

In [11]:

```
1 X_train_dt
```

Out[11]:

	LIGHTING_CONDITION_DARKNESS, LIGHTED ROAD	LIGHTING_CONDITION_DAWN	LIGHT
0	0.0	0.0	
1	1.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	
...	...	...	
82794	0.0	0.0	
82795	1.0	0.0	
82796	0.0	0.0	
82797	0.0	0.0	
82798	1.0	0.0	

82799 rows × 360 columns

1 Complete jupyter nb

1.1 Logistic Regression

1.1.1 Model 2

In [12]:

```
1 lr_2 = LogisticRegression()
2 lr_2.fit(X_train_lr_sel, y_train)
```

```
c:\python\python37\lib\site-packages\sklearn\utils\validation.py
versionWarning: A column-vector y was passed when a 1d array was
lease change the shape of y to (n_samples, ), for example using
  return f(**kwargs)
```

Out[12]:

```
LogisticRegression()
```

1.1.1.1 Model Evaluation

1.1.2 Model 3

In [13]:

```
lr_3 = LogisticRegression()
```

In [14]:

```
params_3 = {"C": [0.001, 0.01, 0.1, 1, 1e7, 1e8, 1e9, 1e10
            "penalty": ["l1", "l2"],
            "solver":["liblinear"]}
```

In [15]:

```
lr_3_gridsearch = GridSearchCV(lr_3, params_3,
                               scoring="recall", n_jobs=-1
lr_3_gridsearch.fit(X_train_lr_sel, y_train)
```

```
c:\python\python37\lib\site-packages\sklearn\utils\validation.py
versionWarning: A column-vector y was passed when a 1d array was
lease change the shape of y to (n_samples, ), for example using
  return f(**kwargs)
```

Out[15]:

```
GridSearchCV(cv=5, estimator=LogisticRegression(), n_jobs=-1,
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10000000.0,
                               1000000000.0, 10000000000.0],
                         'penalty': ['l1', 'l2'], 'solver': ['lil
             scoring='recall')
```

1.1.2.1 Model Evaluation

In [16]:

```
lr_3_gridsearch.best_params_
```

Out[16]:

```
{'C': 1, 'penalty': 'l1', 'solver': 'liblinear'}
```

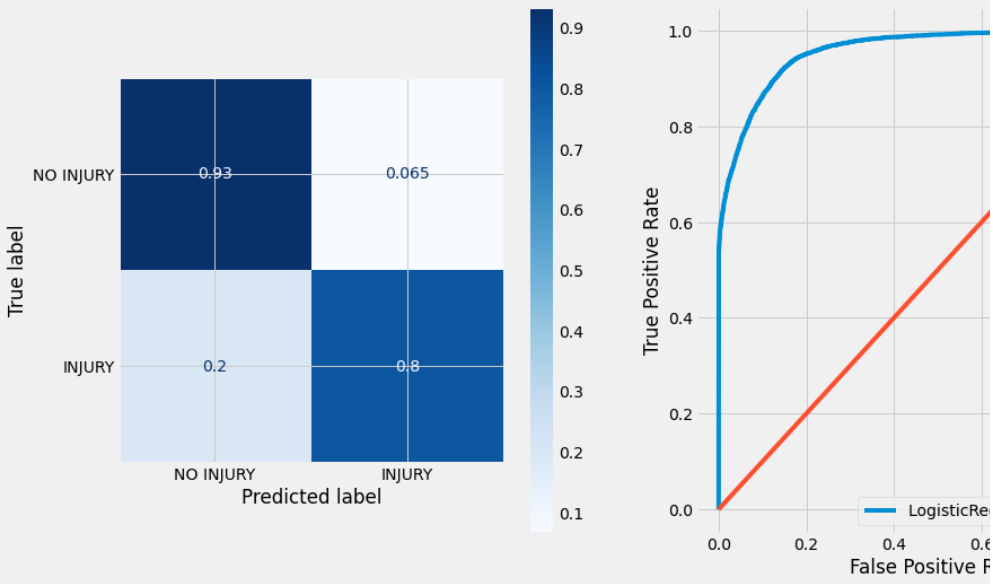
In [17]:

```
1 model_eval(lr_3_gridsearch.best_estimator_, X_train_lr_sel
```

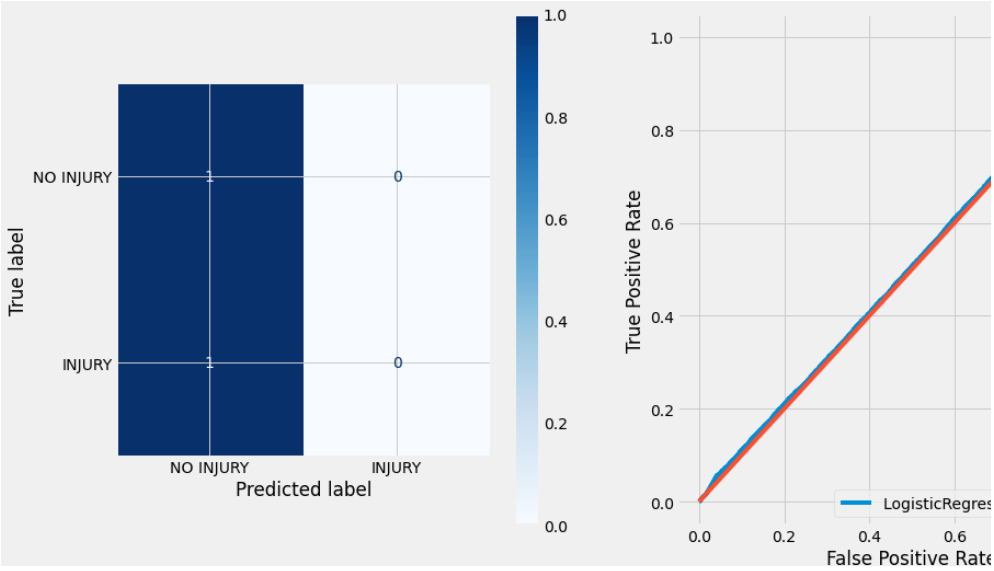
```
MODEL EVAL VS PREVIOUS (TEST)
================================================================
```

	Previous Model	Current Model	Delta
Recall	0	0.8	0.8
F1	0	0.83	0.83
Accuracy	0.67	0.89	0.22
AUC	0.51	0.96	0.45

```
CURRENT MODEL: Not Overfit (Recall)
================================================================


Recall on Training: 0.81
Recall on Test: 0.8


Classification Reports----------------------------------------
              precision    recall  f1-score   support

           0       0.91      0.93      0.92     23838
           1       0.86      0.80      0.83     11648

    accuracy                           0.89     35486
   macro avg       0.88      0.87      0.87     35486
weighted avg       0.89      0.89      0.89     35486

Test Graphs----------------------------------------------------
```




```
PREVIOUS MODEL
================================================================
```




```
<Figure size 432x288 with 0 Axes>
```

> **Observations** Feature selection did not seem to neagtvely impact the m
> feature count by 21%.

> **Actions** See if fizing class imbalance will help the model

1.1.3 Model 4

In [18]:

```
lr_4 = LogisticRegression(class_weight="balanced")
```

In [19]:

```
params_4 = {"C":[.01, 0.1, 1, 1e7, 1e8, 1e9, 1e10],
            "penalty":["l1", "l2"],
            "solver":["liblinear"]}
```

In [20]:

```
1 lr_4_gridsearch = GridSearchCV(lr_4, params_4, scoring="re
2 lr_4_gridsearch.fit(X_train_lr_sel, y_train)
```

```
c:\python\python37\lib\site-packages\sklearn\utils\validation.py
versionWarning: A column-vector y was passed when a 1d array was
lease change the shape of y to (n_samples, ), for example using
  return f(**kwargs)
```

Out[20]:

```
GridSearchCV(cv=5, estimator=LogisticRegression(class_weight='ba
             n_jobs=-1,
             param_grid={'C': [0.01, 0.1, 1, 10000000.0, 1000000
                               1000000000.0, 10000000000.0],
                         'penalty': ['l1', 'l2'], 'solver': ['li
             scoring='recall')
```

In [21]:

```
1 lr_4_gridsearch.best_params_
```

Out[21]:

```
{'C': 0.01, 'penalty': 'l1', 'solver': 'liblinear'}
```

1.1.3.1 Model Evaluation

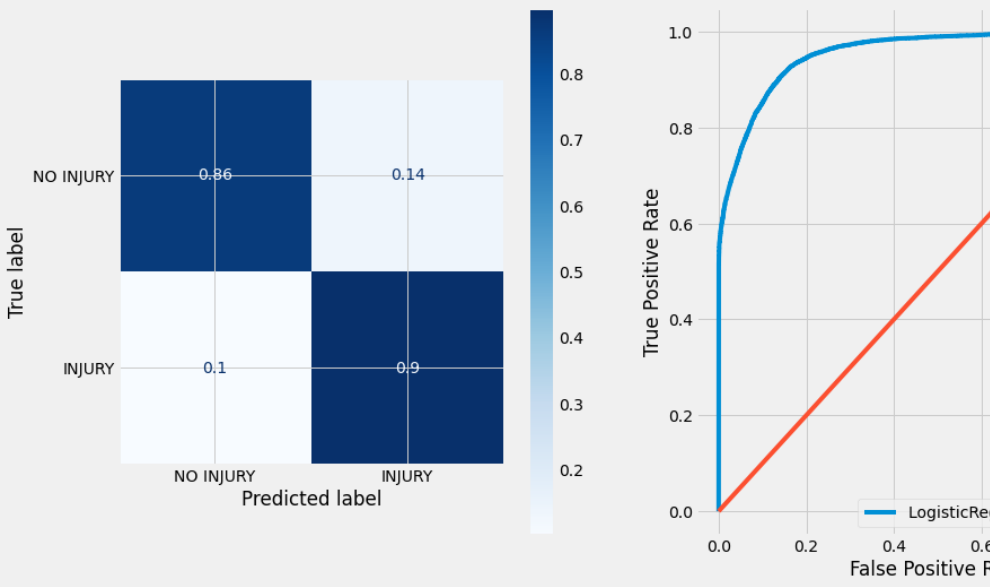
In [22]:

```
1 model_eval(lr_4_gridsearch.best_estimator_, X_train_lr_sel
2           X_test_lr_sel,y_test,
3            prev_model=lr_3_gridsearch.best_estimator_ )
```

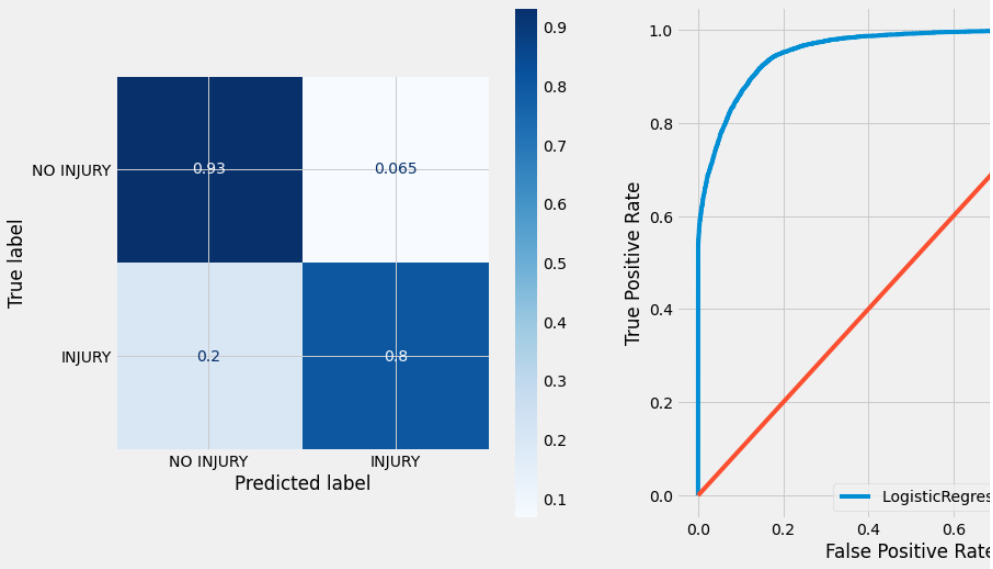
```
MODEL EVAL VS PREVIOUS (TEST)
================================================================
```

	Previous Model	Current Model	Delta
Recall	0.8	0.9	0.1
F1	0.83	0.83	0
Accuracy	0.89	0.88	-0.01
AUC	0.96	0.96	0

```
CURRENT MODEL: Not Overfit (Recall)
================================================================


Recall on Training: 0.9
Recall on Test: 0.9


Classification Reports----------------------------------------
              precision    recall  f1-score   support

           0       0.95      0.86      0.90     23838
           1       0.76      0.90      0.83     11648

    accuracy                           0.88     35486
   macro avg       0.86      0.88      0.86     35486
weighted avg       0.89      0.88      0.88     35486

Test Graphs----------------------------------------------------
```




```
PREVIOUS MODEL
================================================================
```




```
<Figure size 432x288 with 0 Axes>
```

> **Observations** Balancing the classes improved the model by increasing
> and increased the accuracy by 1%

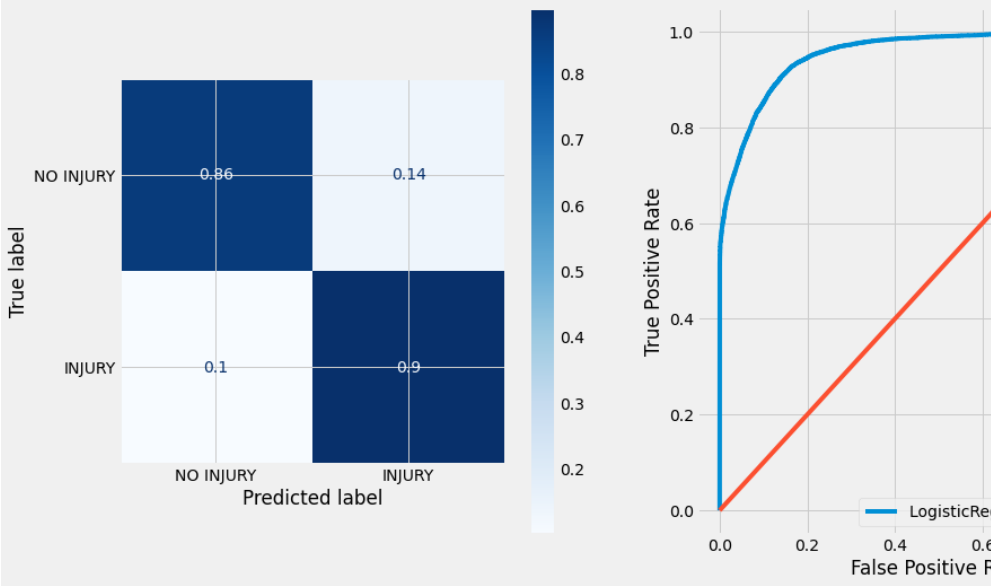
1.1.4 Best LR Model

The besst Logistic Regression model was model #4 which produced the followin
important features:

In [46]:

```
model_eval(lr_4_gridsearch.best_estimator_,
           X_train_lr_sel, y_train, X_test_lr_sel,
           y_test)
```

```
CURRENT MODEL: Not Overfit (Recall)
========================================================================


Classification Reports----------------------------------------
              precision    recall  f1-score   support

           0       0.95      0.86      0.90     23838
           1       0.76      0.90      0.83     11648

    accuracy                           0.88     35486
   macro avg       0.86      0.88      0.86     35486
weighted avg       0.89      0.88      0.88     35486

Test Graphs----------------------------------------------------
```



In [52]:

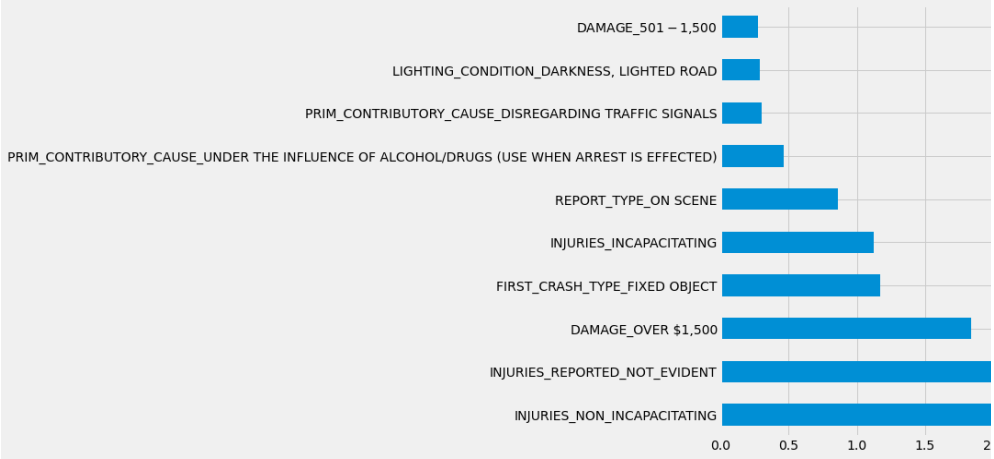
```
#find top coefficients
best_coeffs = get_coefficients(lr_4_gridsearch.best_estima
best_coeffs_pos = best_coeffs.loc[best_coeffs >= 0].sort_v
best_coeffs_neg = best_coeffs.loc[best_coeffs < 0].sort_va
best_coeffs_overall = best_coeffs.abs().sort_values(ascend
```

In [56]:

```
1 fig = plt.subplots(figsize=(8, 8))
2 best_coeffs_pos[:10].plot.barh()
```

Out[56]:

<AxesSubplot:>

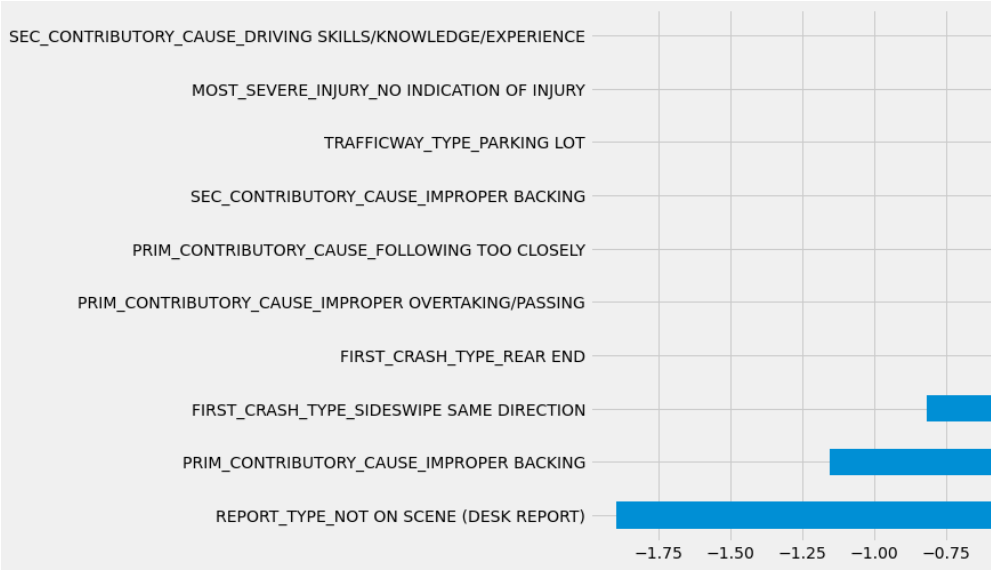


In [53]:

```
1 fig = plt.subplots(figsize=(8, 8))
2 best_coeffs_neg[:10].plot.barh()
```

Out[53]:

<AxesSubplot:>



1.2 Random Forest

1.2.1 Model 1

In [23]:

```
1 X_train_dt
```

Out[23]:

	LIGHTING_CONDITION_DARKNESS, LIGHTED ROAD	LIGHTING_CONDITION_DAWN	LIGHT
0	0.0	0.0	
1	1.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	
...	...	...	
82794	0.0	0.0	
82795	1.0	0.0	
82796	0.0	0.0	
82797	0.0	0.0	
82798	1.0	0.0	

82799 rows × 360 columns

In [24]:

```
1 rf_1 = RandomForestClassifier()
2 rf_1.fit(X_train_dt, y_train)
```

```
c:\python\python37\lib\site-packages\ipykernel_launcher.py:2: Da
Warning: A column-vector y was passed when a 1d array was expecte
hange the shape of y to (n_samples,), for example using ravel().
```

Out[24]:

```
RandomForestClassifier()
```

1.2.1.1 Model Evaluation

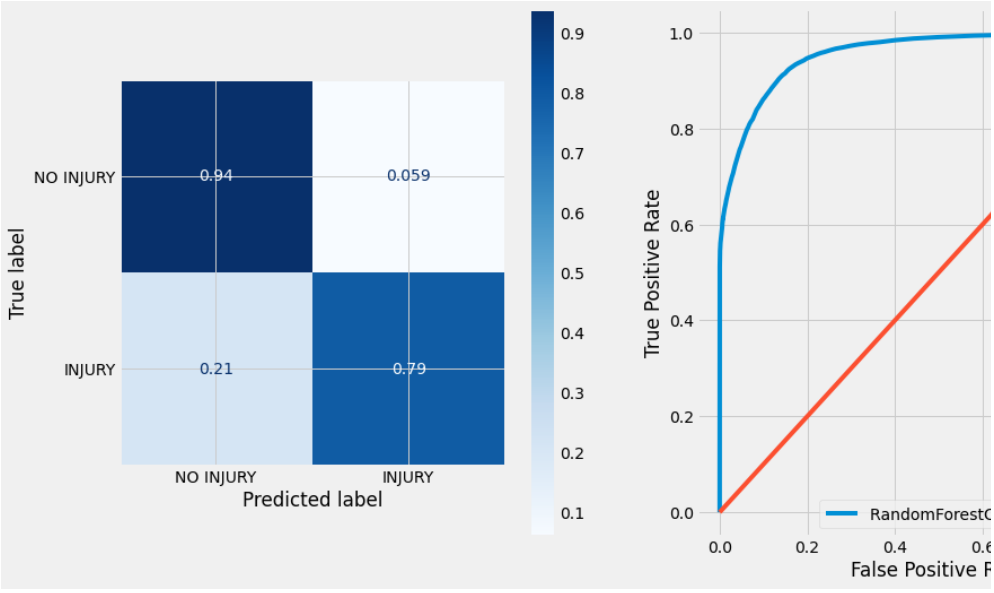
In [25]:

```
model_eval(rf_1, X_train_dt, y_train, X_test_dt, y_test,
           prev_model=lr_4_gridsearch.best_estimator_,
          prev_X_train=X_train_lr_sel,
          prev_y_train=y_train,
           prev_X_test=X_test_lr_sel, prev_y_test=y_test)
```

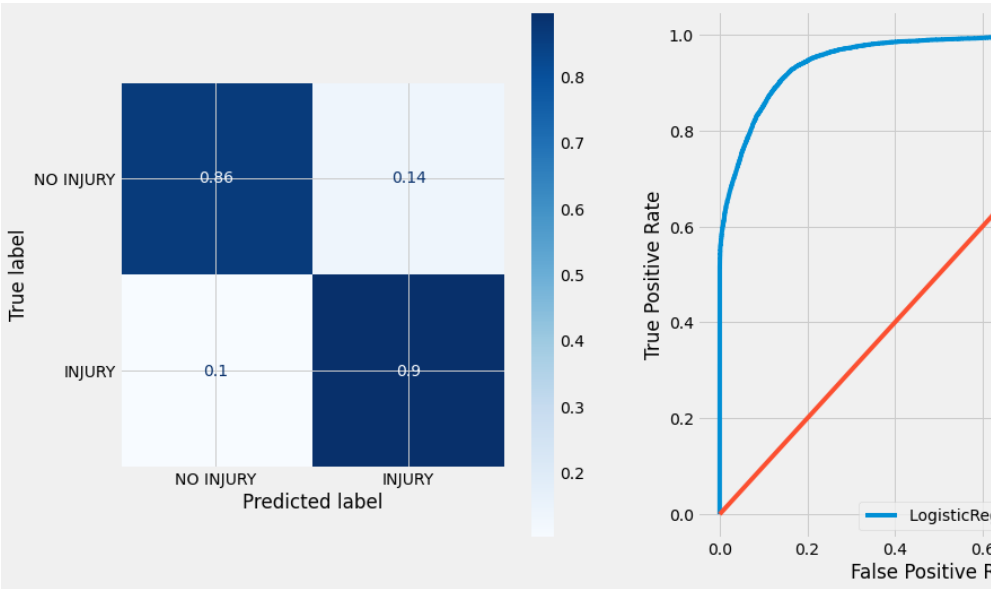
```
MODEL EVAL VS PREVIOUS (TEST)
================================================================
```

	Previous Model	Current Model	Delta
Recall	0.9	0.79	-0.11
F1	0.83	0.83	0
Accuracy	0.88	0.89	0.01
AUC	0.96	0.96	0

```
CURRENT MODEL: Overfit (Recall)
================================================================


Recall on Training: 1.0
Recall on Test: 0.79


Classification Reports----------------------------------------
              precision    recall  f1-score   support

           0       0.90      0.94      0.92     23838
           1       0.87      0.79      0.83     11648

    accuracy                           0.89     35486
   macro avg       0.88      0.87      0.87     35486
weighted avg       0.89      0.89      0.89     35486

Test Graphs---------------------------------------------------
```



```
PREVIOUS MODEL
================================================================
```




```
<Figure size 432x288 with 0 Axes>
```

1.2.2 Model 2

In [26]:

```
1 rf_2 = RandomForestClassifier()
```

In [27]:

```
params = [{"n_estimators":[100, 500, 1000],
           "criterion":["entropy", "gini"],
           "max_features":["auto", "log2"],
           "class_weight":["balanced"],
           "max_depth":[20, 35, 65],
           "min_samples_split":[2, 3],
           "min_samples_leaf":[1, 2, 3]}]
```

In [28]:

```
rf_2_gridsearch = GridSearchCV(rf_2, params, scoring="reca
rf_2_gridsearch.fit(X_train_dt, y_train)
```

```
c:\python\python37\lib\site-packages\sklearn\model_selection\_sea
DataConversionWarning: A column-vector y was passed when a 1d ar
cted. Please change the shape of y to (n_samples,), for example
().
  self.best_estimator_.fit(X, y, **fit_params)
```

Out[28]:

```
GridSearchCV(cv=3, estimator=RandomForestClassifier(), n_jobs=-1
             param_grid=[{'class_weight': ['balanced'],
                          'criterion': ['entropy', 'gini'],
                          'max_depth': [20, 35, 65],
                          'max_features': ['auto', 'log2'],
                          'min_samples_leaf': [1, 2, 3],
                          'min_samples_split': [2, 3],
                          'n_estimators': [100, 500, 1000]}],
             scoring='recall')
```

1.2.2.1 Model Evaluation

In [29]:

```
rf_2_gridsearch.best_params_
```

Out[29]:

```
{'class_weight': 'balanced',
 'criterion': 'entropy',
 'max_depth': 20,
 'max_features': 'auto',
 'min_samples_leaf': 3,
 'min_samples_split': 2,
 'n_estimators': 100}
```

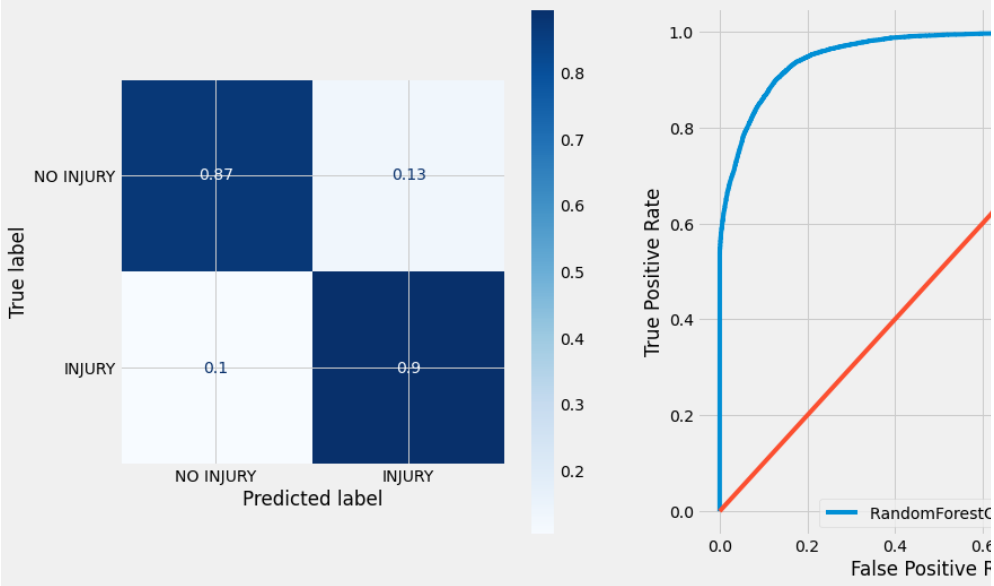
In [30]:

```
1 model_eval(rf_2_gridsearch.best_estimator_, X_train_dt, y_
2           prev_model=rf_1)
```

```
MODEL EVAL VS PREVIOUS (TEST)
================================================================
```

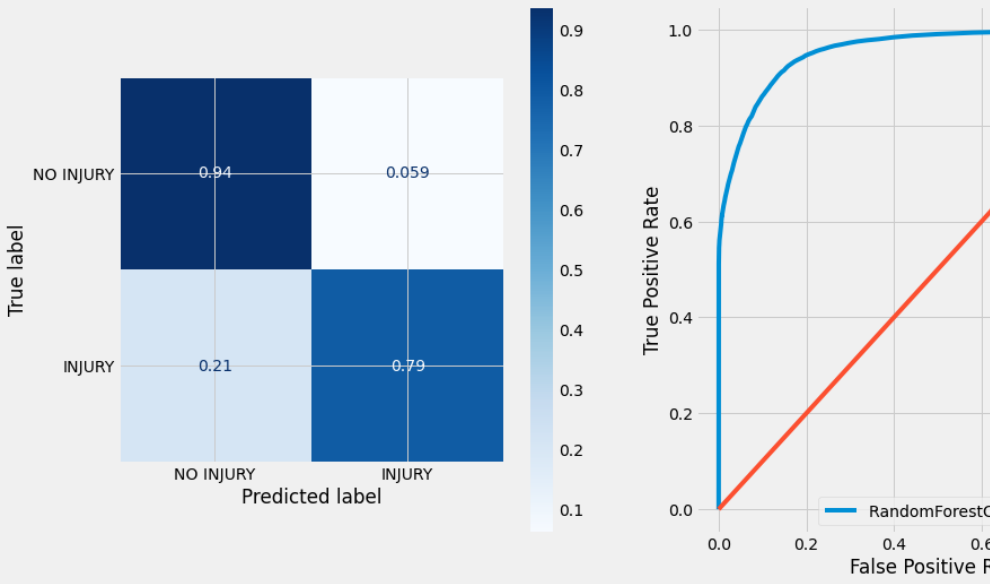
	Previous Model	Current Model	Delta
Recall	0.79	0.9	0.11
F1	0.83	0.83	0
Accuracy	0.89	0.88	-0.01
AUC	0.96	0.96	0

```
CURRENT MODEL: Not Overfit (Recall)
================================================================


Recall on Training: 0.91
Recall on Test: 0.9


Classification Reports----------------------------------------
              precision    recall  f1-score   support

           0       0.95      0.87      0.91     23838
           1       0.78      0.90      0.83     11648

    accuracy                           0.88     35486
   macro avg       0.86      0.89      0.87     35486
weighted avg       0.89      0.88      0.88     35486

Test Graphs----------------------------------------------------
```



Contents


```
PREVIOUS MODEL
================================================================================
```




```
<Figure size 432x288 with 0 Axes>
```

Observations

- The model increased recall by 12% but decreased accucacy by 1%

In [33]:

```
1 rf_2_fi = rf_2_gridsearch.best_estimator_.feature_importan
```

In [42]:

```
1 rf_2_features = pd.DataFrame(data=rf_2_fi, index=X_train_d
2                              columns=["Feature Importances"
3 rf_2_features_sorted = rf_2_features[:10].sort_values("Fea
4
5 rf_2_features_sorted
```

Out[42]:

	Feature I
LIGHTING_CONDITION_DARKNESS, LIGHTED ROAD	
LIGHTING_CONDITION_DAYLIGHT	
SEC_CONTRIBUTORY_CAUSE_DISREGARDING TRAFFIC SIGNALS	
LIGHTING_CONDITION_DAWN	
LIGHTING_CONDITION_DUSK	
SEC_CONTRIBUTORY_CAUSE_DISREGARDING STOP SIGN	
SEC_CONTRIBUTORY_CAUSE_DISTRACTION - FROM INSIDE VEHICLE	
SEC_CONTRIBUTORY_CAUSE_DISREGARDING OTHER TRAFFIC SIGNS	
SEC_CONTRIBUTORY_CAUSE_DISREGARDING ROAD MARKINGS	
SEC_CONTRIBUTORY_CAUSE_CELL PHONE USE OTHER THAN TEXTING	

In [37]:

```
1 rf_2_features_sorted
```

Out[37]:

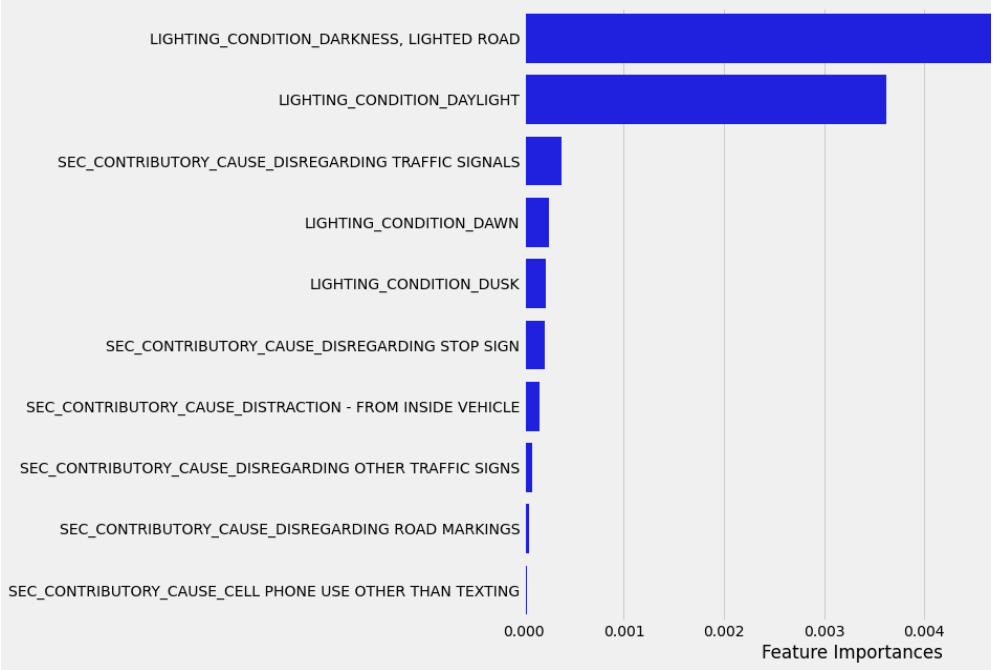
	Feature I
LIGHTING_CONDITION_DARKNESS, LIGHTED ROAD	
LIGHTING_CONDITION_DAYLIGHT	
SEC_CONTRIBUTORY_CAUSE_DISREGARDING TRAFFIC SIGNALS	
LIGHTING_CONDITION_DAWN	
LIGHTING_CONDITION_DUSK	
SEC_CONTRIBUTORY_CAUSE_DISREGARDING STOP SIGN	
SEC_CONTRIBUTORY_CAUSE_DISTRACTION - FROM INSIDE VEHICLE	
SEC_CONTRIBUTORY_CAUSE_DISREGARDING OTHER TRAFFIC SIGNS	
SEC_CONTRIBUTORY_CAUSE_DISREGARDING ROAD MARKINGS	
SEC_CONTRIBUTORY_CAUSE_CELL PHONE USE OTHER THAN TEXTING	

In [39]:

```
1  fig, ax= plt.subplots(figsize=(10, 10))
2  sns.barplot(data=rf_2_features_sorted,
3              x=rf_2_features_sorted["Feature Importances"],
4              y = rf_2_features_sorted.index, orient="h", col
```

Out[39]:

```
<AxesSubplot:xlabel='Feature Importances'>
```



Contents

2 Data Interpretation

> This dataset took a lot of cleaning, especially involving unknown values
> original dataset. After cleaning, a logistic regression and random forest
> trained to predict civilians in Chicago being injured or not. Classification
> and optimized for finding an injured civilian due to car crash.

In [ ]:

```
1
```

Contents