Disputed Workers' Compensation Claims in New York State

Crystal Bevis

Background:

- Workers' compensation claims sent to the New York State Workers' Compensation Board (WCB)
- About 5% of claims have are disputed (controverted) by insurance companies
  - Unbalanced data set
- Some information we have:
  - Part of body injured, type of injury, cause of injury
  - Geographical region
  - Weekly salary of employee
  - Dates various forms are recieved by WCB

Objective:

- Predict which Workers' Compensation Board claims were controverted by the insurance companies.
- Ultimately could help determine which claims should have more resources allocated to them upfront.

Data Sources and References:

NYS Workers' Compensation Board Data - Over 1 million data points:
https://www.kaggle.com/new-york-state/nys-assembled-workers'-compensation-claims (https://www.kaggle.com/new-york-state/nys-assembled-workers'-compensation-claims)
https://data.ny.gov/Government-Finance/Assembled-Workers-Compensation-Claims-Beginning-20/jshw-gkgu (https://data.ny.gov/Government-Finance/Assembled-Workers-Compensation-Claims-Beginning-20/jshw-gkgu)

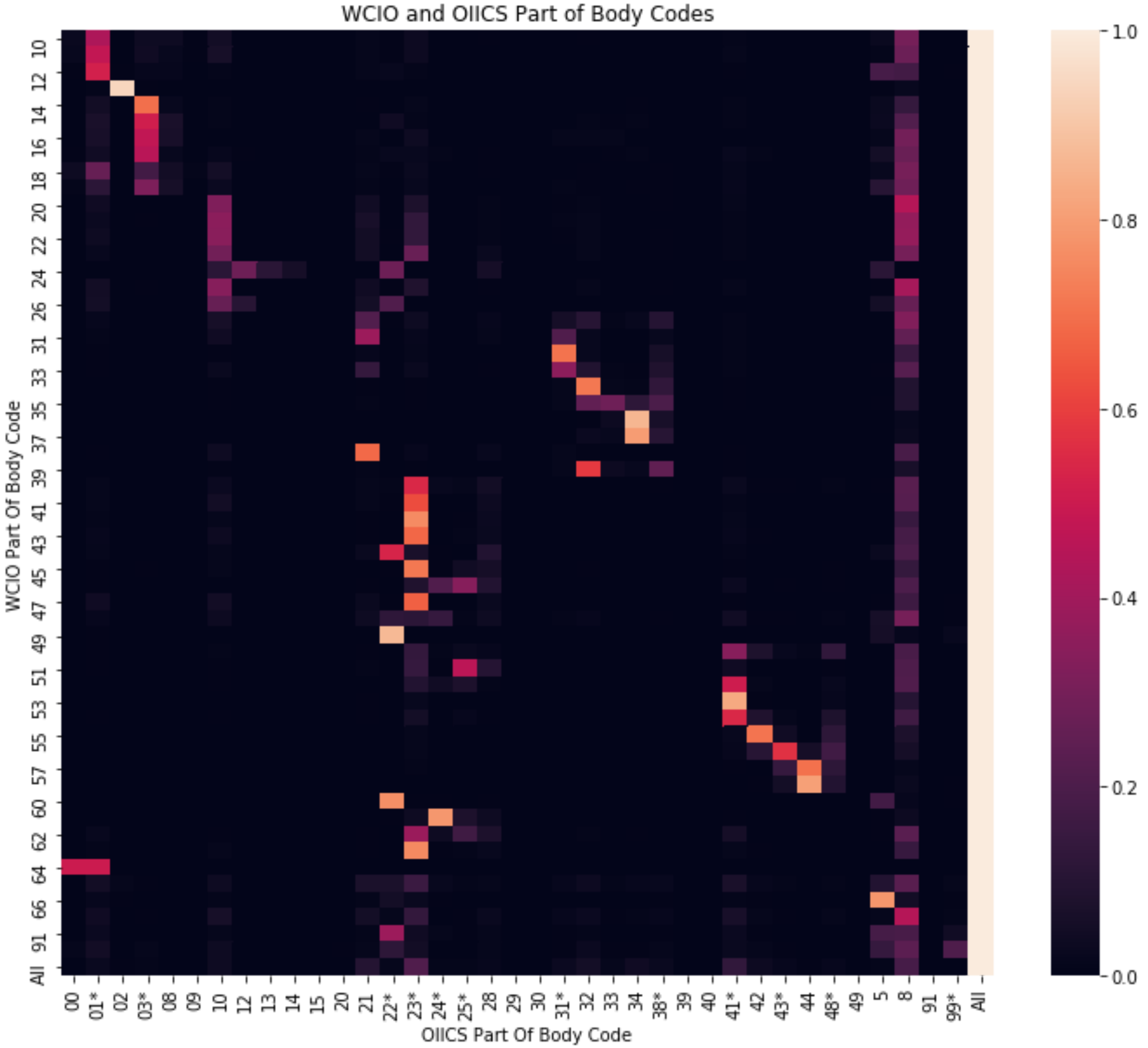
OIICS Codes (appears version 1.01 used):
https://wwwn.cdc.gov/wisards/oiics/Trees/MultiTree.aspx?Year=2007 (https://wwwn.cdc.gov/wisards/oiics/Trees/MultiTree.aspx?Year=2007)

Claims Process:
http://www.wcb.ny.gov/content/main/onthejob/HowSystemWorks.jsp (http://www.wcb.ny.gov/content/main/onthejob/HowSystemWorks.jsp)

New York State Regions - Combine counties into regions:
https://en.wikipedia.org/wiki/Category:Regions_of_New_York_(state (https://en.wikipedia.org/wiki/Category:Regions_of_New_York_(state))

Exploratory Data Analysis

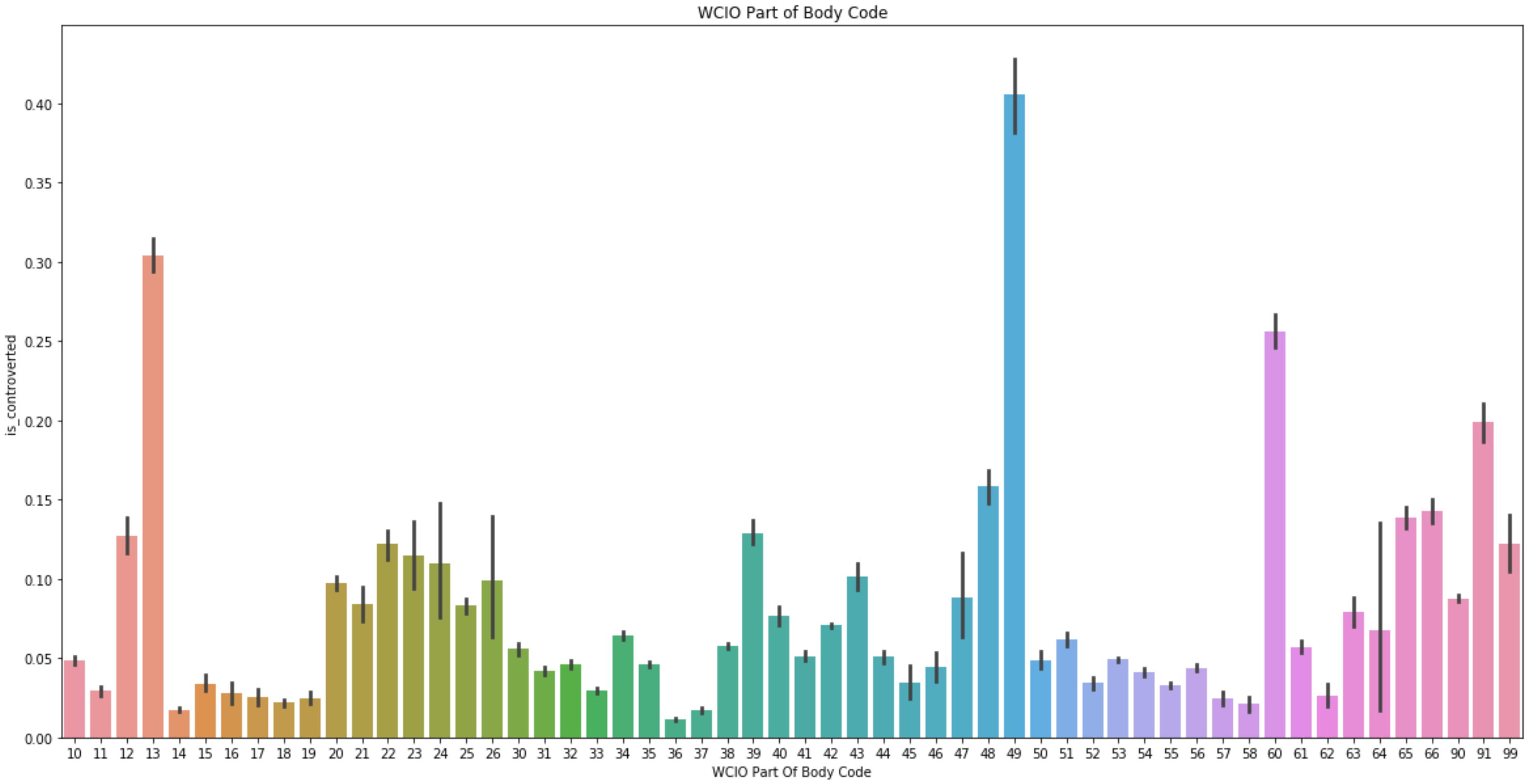
- Two different sets of injury codes WCIO and OIICS
- Cause of injury correlated with type of injury? Redundant features?



WCIO and OIICS Part of Body Codes
WCIO Part Of Body Code
OIICS Part Of Body Code
10
12
14
16
18
20
22
24
26
31
33
35
37
39
41
43
45
47
49
51
53
55
57
60
62
64
66
91
All
00
01*
02
03*
08
09
10
12
13
14
15
20
21
22*
23*
24*
25*
28
29
30
31*
32
33
34
38*
39
40
41*
42
43*
44
48*
49
5
8
91
99*
All
1.0
0.8
0.6
0.4
0.2
0.0

In [13]:

```
plt.figure(figsize=(20, 10))
sns.barplot(x='WCIO Part Of Body Code', y='is_controverted', data=df)
plt.title('WCIO Part of Body Code')
plt.show()
```



In [18]:
```
sns.barplot(x='Gender', y='Average Weekly Wage', hue='is_controverted', data=df)
plt.title('Gender, Wages, and Controverted')
plt.show()
```

Modeling - Selected Features

- WCIO codes: more fields filled out
- Omitted age: too many NaNs
- Used accident and assembly dates:
  - Not time to ANCR (difference between those dates)
  - Time of year more important than difference between dates
- Gender
- Region
- Type of insurance
  - Not insurance carrier - too many unique carriers

Model 1: Random Forest

- Fits a number of decision tree classifiers on sub-samples of the dataset.
- Less prone to overfitting than a decision tree
- Label encoded data set
- Look at feature importances
- For handling the unbalanced dataset, try regular, under sampled, and SMOTE
  - Regular sampling - good accuracy, poor recall
  - Under sampled and SMOTE, overfit to data, high recall, low precision

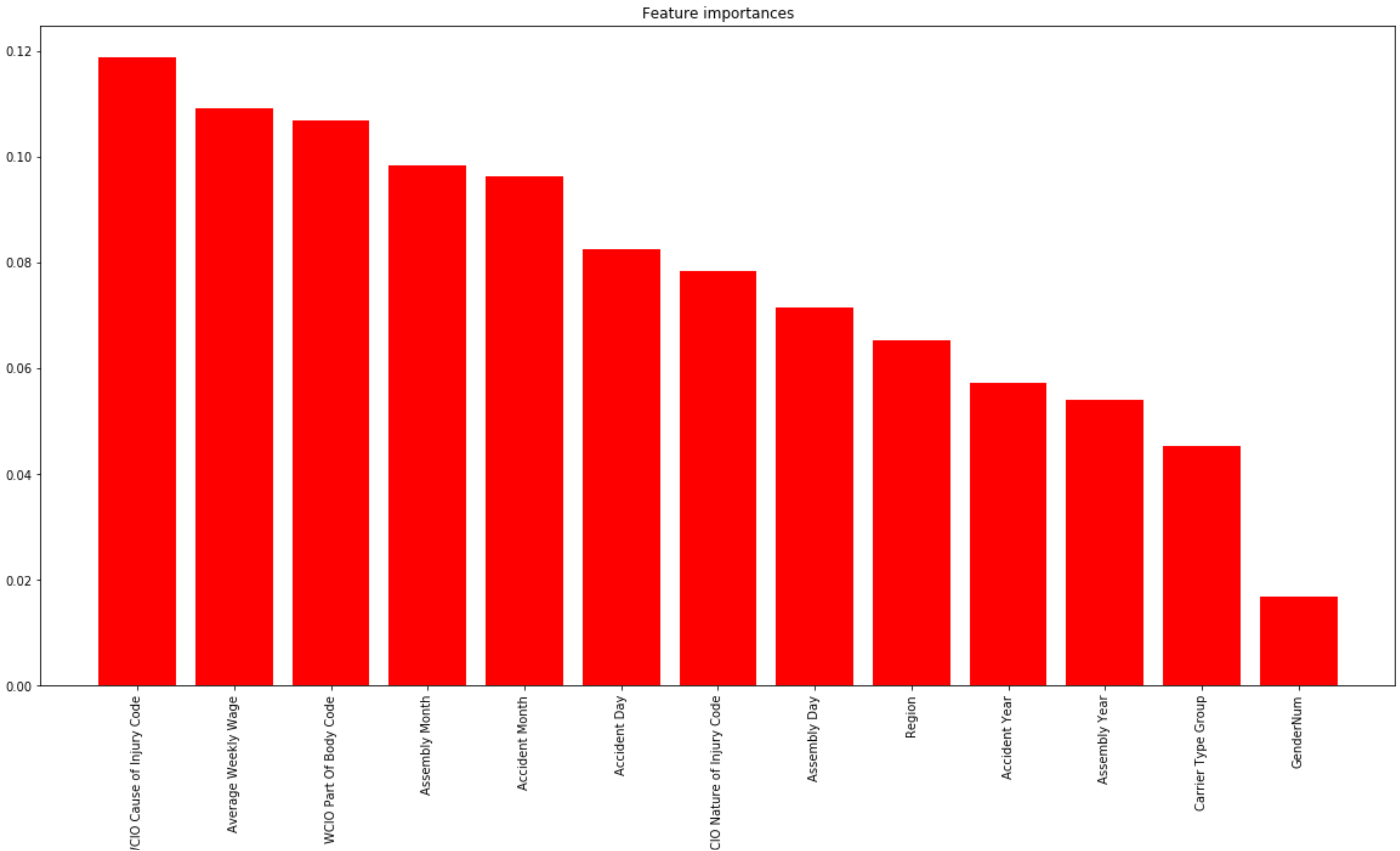
Results

- Score: 0.743
- Precision: 0.141
- Recall: 0.696

In [52]:

```
importances = rfc.feature_importances_
indices = np.argsort(importances)[::-1]

# Plot the feature importances of the forest
plt.figure(figsize=(20,10))
plt.title("Feature importances")
plt.bar(range(len(indices)), importances[indices], color="r", align="center")
plt.xticks(range(len(indices)), X.columns[indices], rotation=90)
plt.xlim([-1, len(indices)])
plt.show()
```

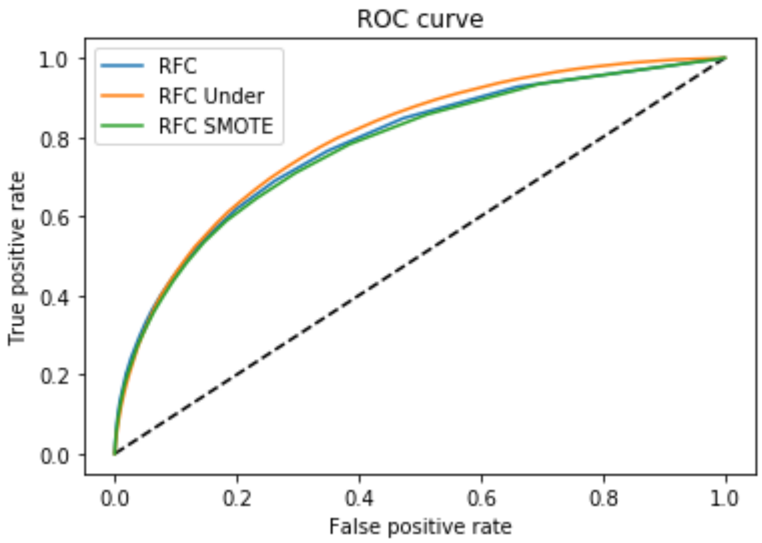


In [55]:

```
# Look at the ROC curves for each sampling
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
fpr_rfc, tpr_rfc, _ = roc_curve(y_test, y_pred_prob_rfc)
fpr_rfc_under, tpr_rfc_under, _ = roc_curve(y_test, y_pred_prob_rfc_under)
fpr_rfc_SMOTE, tpr_rfc_SMOTE, _ = roc_curve(y_test, y_pred_prob_rfc_SMOTE)

plt.plot(fpr_rfc, tpr_rfc, label='RFC')
plt.plot(fpr_rfc_under, tpr_rfc_under, label='RFC Under')
plt.plot(fpr_rfc_SMOTE, tpr_rfc_SMOTE, label='RFC SMOTE')

plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```



Model 2: Gradient Boost Classifier

- Tried regular, under, and SMOTE sampling of data
- Under and regular performed about the same
    - Would use under sampled - faster
- Slower than random forest

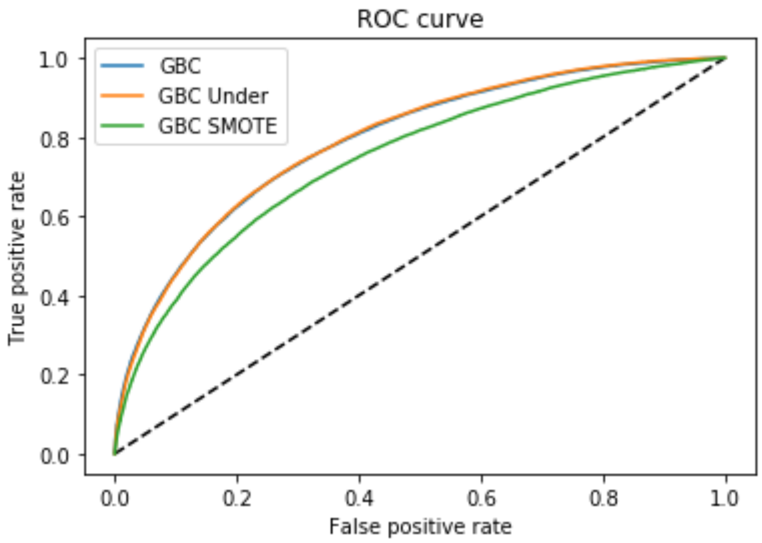
Results:

- Accuracy Score: 0.746
- Recall Score: 0.684
- Precision Score: 0.141

In [59]:
```
# Look at the ROC curves for each sampling
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
fpr_gbc, tpr_gbc, _ = roc_curve(y_test, y_pred_prob_gbc)
fpr_gbc_under, tpr_gbc_under, _ = roc_curve(y_test, y_pred_prob_gbc_under)
fpr_gbc_SMOTE, tpr_gbc_SMOTE, _ = roc_curve(y_test, y_pred_prob_gbc_SMOTE)

plt.plot(fpr_gbc, tpr_gbc, label='GBC')
plt.plot(fpr_gbc_under, tpr_gbc_under, label='GBC Under')
plt.plot(fpr_gbc_SMOTE, tpr_gbc_SMOTE, label='GBC SMOTE')

plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```



Light Gradient Boosting

- Grows models leaf-wise instead of depth
- Computationally less expensive than Gradient Boosting
- Can handle categorical variables label encoded
- Used regular dataset, undersampled, and SMOTE
- All three performed about the same, use the under sampled data because it is less computationally expensive

Scores

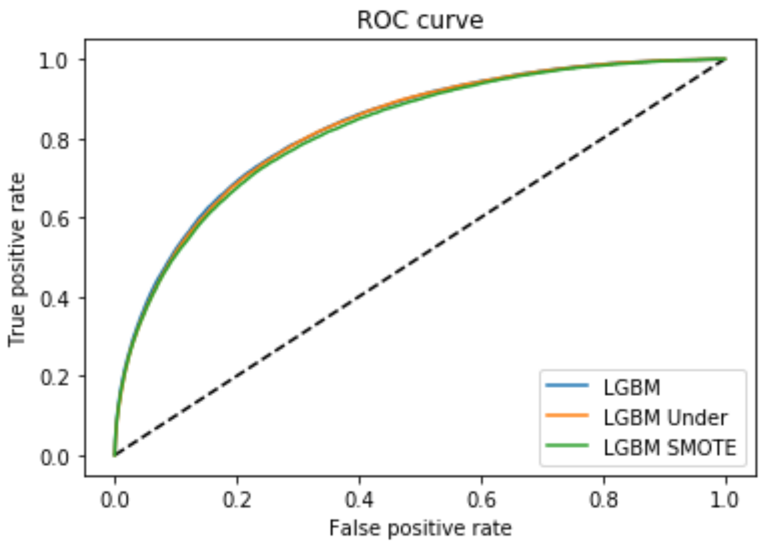
- Accuracy Score LightGBM: 0.755
- Recall Score: 0.733
- Precision Score: 0.155

In [64]:

```
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
fpr_lgbm, tpr_lgbm, _ = roc_curve(y_test, y_pred_proba_lgbm)
fpr_lgbm_under, tpr_lgbm_under, _ = roc_curve(y_test, y_pred_proba_lgbm_under)
fpr_lgbm_SMOTE, tpr_lgbm_SMOTE, _ = roc_curve(y_test, y_pred_proba_lgbm_SMOTE)

plt.plot(fpr_lgbm, tpr_lgbm, label='LGBM')
plt.plot(fpr_lgbm_under, tpr_lgbm_under, label='LGBM Under')
plt.plot(fpr_lgbm_SMOTE, tpr_lgbm_SMOTE, label='LGBM SMOTE')

plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```



Logistic Regression

- One hot encode categorical features
- Lasso regression: number of features is > 200 after one hot encoding
- SMOTE does not work well on this number of features
    - Slow, many NAN with one hot categorical
    - also tried using SMOTENC - too many categorical features
- Try undersampling majority class and using all data points
- Select K Best to reduce features ahead of times

Scores

- Logistic Regression Under Sampled Score: 0.721
- Logistic Regression Under Sampled precision score 0.129
- Logistic Regression Under Sampled recall score 0.681

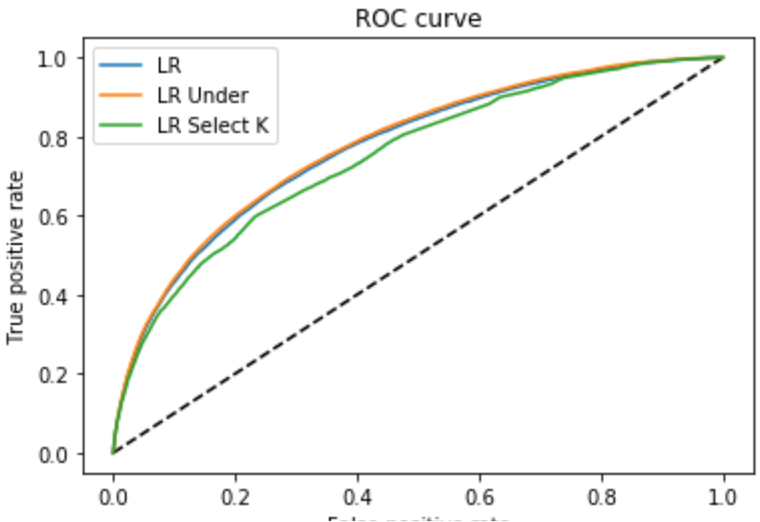
In [81]:

```
# Look at the ROC curves for each of the three forest models to see which of the s
ampling methods
# made the biggest difference or if they are the same

plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')
fpr_lr, tpr_lr, _ = roc_curve(y_test_one, y_pred_prob_lr)
fpr_lr_under, tpr_lr_under, _ = roc_curve(y_test_one, y_pred_prob_lr_under)
fpr_lr_k, tpr_lr_k, _ = roc_curve(y_test2, y_pred_prob_lr_k)
#fpr_lr_SMOTE, tpr_lr_SMOTE, _ = roc_curve(y_test, y_pred_prob_lr_SMOTE)

plt.plot(fpr_lr, tpr_lr, label='LR')
plt.plot(fpr_lr_under, tpr_lr_under, label='LR Under')
plt.plot(fpr_lr_k, tpr_lr_k, label='LR Select K')
#plt.plot(fpr_lr_SMOTE, tpr_lr_SMOTE, label='LR SMOTE')

plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```



Model Selection

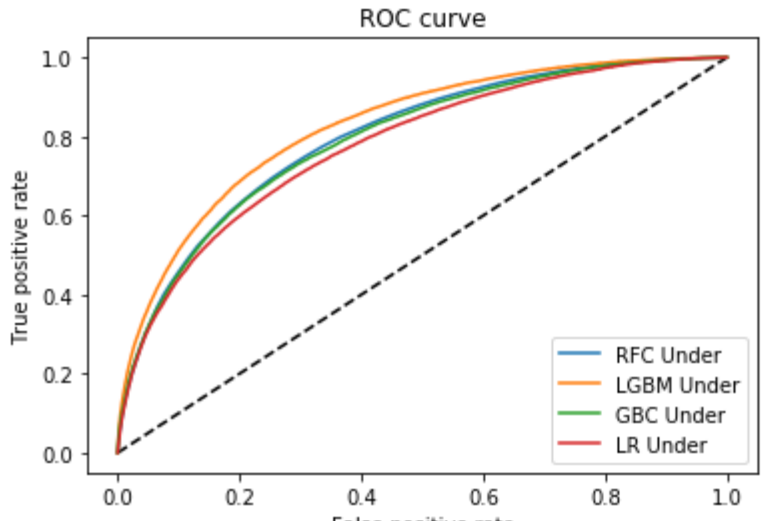
In [82]:

```
plt.figure(1)
plt.plot([0, 1], [0, 1], 'k--')

# Use the unique names for y_test here to make sure they don't plot incorrectly
fpr_rfc_under, tpr_rfc_under, _ = roc_curve(y_test_label, y_pred_prob_rfc_under)
fpr_lr_under, tpr_lr_under, _ = roc_curve(y_test_one, y_pred_prob_lr_under)
fpr_lgbm_under, tpr_lgbm_under, _ = roc_curve(y_test_label, y_pred_proba_lgbm_unde
r)
fpr_gbc_under, tpr_gbc_under, _ = roc_curve(y_test_label, y_pred_prob_gbc_under)

plt.plot(fpr_rfc_under, tpr_rfc_under, label='RFC Under')
plt.plot(fpr_lgbm_under, tpr_lgbm_under, label='LGBM Under')
plt.plot(fpr_gbc_under, tpr_gbc_under, label='GBC Under')
plt.plot(fpr_lr_under, tpr_lr_under, label='LR Under')


plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC curve')
plt.legend(loc='best')
plt.show()
```



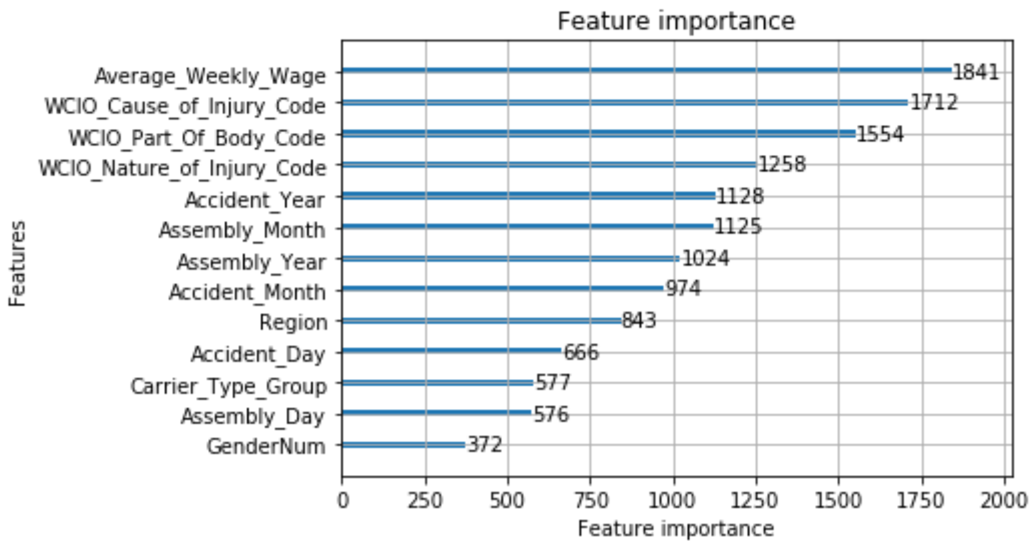
Cross Validation LGBM

- Check what the log-loss is when training set split into 5 folds
- Looked at STD to see if much deviation between the log-loss
    - Found a low STD (.006), the difference between the log-loss in folds was low
- Use to determine over-fit to part of the training set
- Early stopping

In [93]:

```
# Plot the feature importances
print('Feature Importances')
ax = lgbm.plot_importance(clf_under, max_num_features=20)
plt.show()
```

```
Feature Importances
```



Optimize Parameters for LightGBM

- Set up a grid search
- Did larger ranges for parameters at different times for computational reasons
- Narrowed down to different ranges
- Focused on sub_feature, min_data_leaf, bagging_fraction, bagging_freq, learning_rate, num_leaves

Model Shortcomings:

- Expert in the field could determine other usable features
- Recall comes at high precision cost
- Only optimized the best model
- Dates are not treated cyclically
  - January 1, December 12
- SMOTE implementation
  - Better for continuous variables

Conclusion:

- Best Model: Light Gradient Boost Model
  - Best area under receiver operating characteristic
  - Fast
- Best Sampling Method:
  - Under Sampling
- Potential Future Projects:
  - Predict which cases go to Supreme Court
  - Look into regions of New York with high rates of claims, why is that?
  - Cluster analysis into claims
- If you don't want your insurance claim controverted:
  - Puncture your great toe in May in the Finger Lakes region of New York State