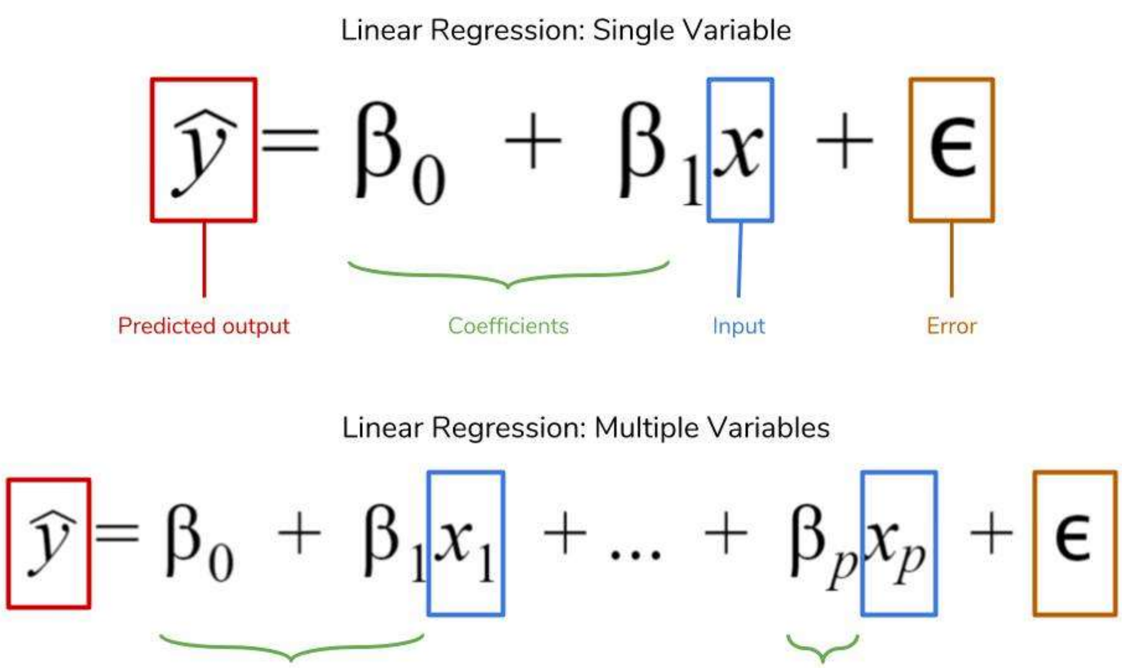
Use of Linear and Logistic Regression Coefficients with Lasso (L1) and Ridge (L2) Regularization for Feature Selection in Machine Learning

Linear Regression



Basic Assumptions

- Linear relationship with the target y
- Feature space X should have gaussian distribution
- Features are not correlated with other
- Features are in same scale i.e. have same variance

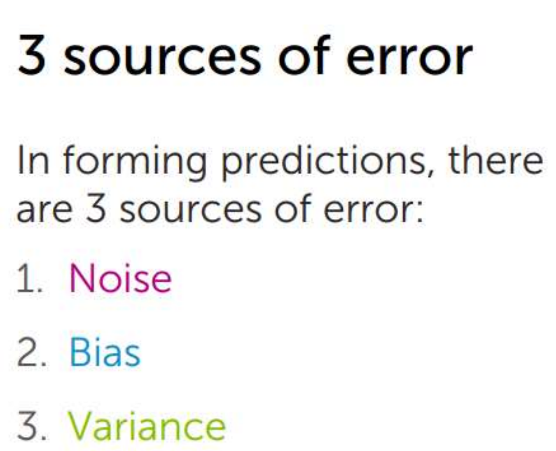
Lasso (L1) and Ridge (L2) Regularization

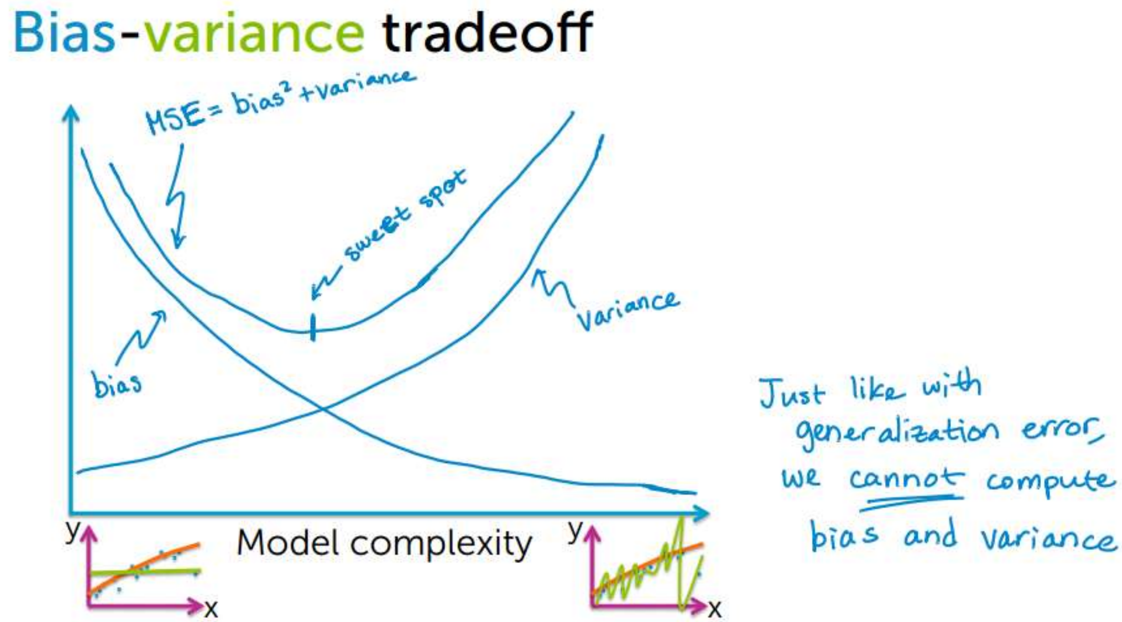
Regularization is a technique to discourage the complexity of the model. It does this by penalizing the loss function. This helps to solve the overfitting problem.

- L1 regularization (also called Lasso)
- L2 regularization (also called Ridge)
- L1/L2 regularization (also called Elastic net)

A regression model that uses L1 regularization technique is called Lasso Regression and model which uses L2 is called Ridge Regression.

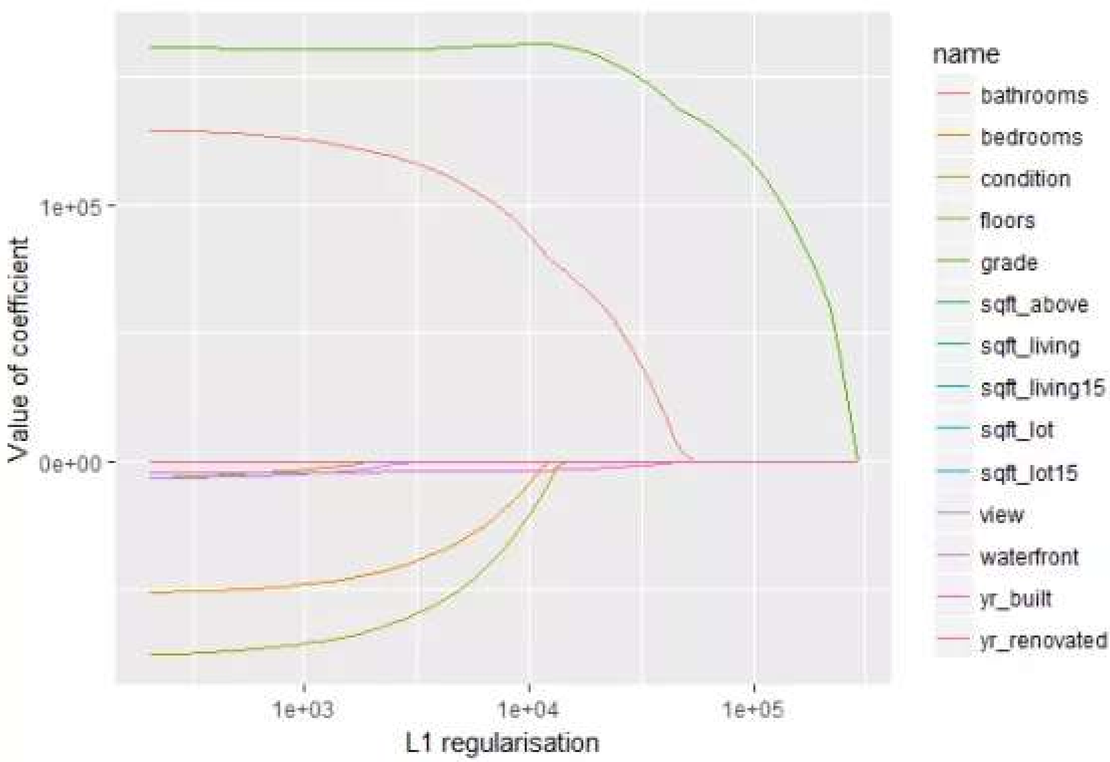
What is Lasso Regularisation

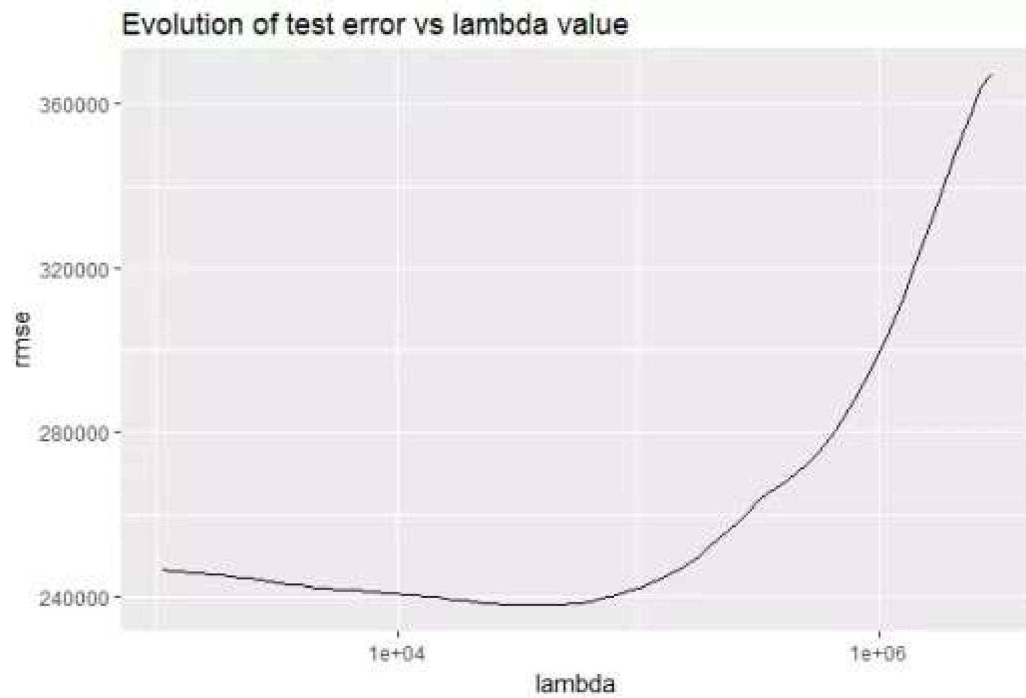




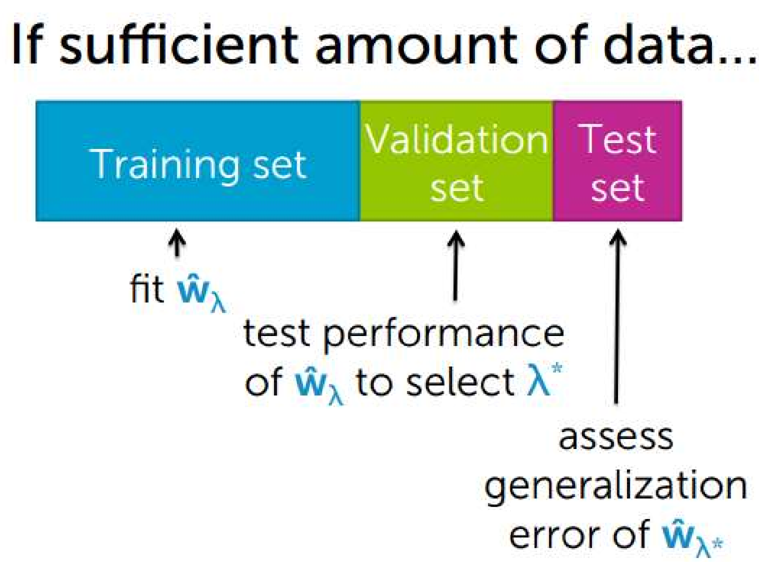
$$\text{RSS}(\mathbf{w}) + \lambda||\mathbf{w}||_1 = \sum_{i=1}^{N}(y_i - w_0 h_0(\mathbf{x}_i) - w_1 h_1(\mathbf{x}_i))^2 + \lambda\,(|w_0|+|w_1|)$$

The L1 regularization adds a penalty equal to the sum of the absolute value of the coefficients.



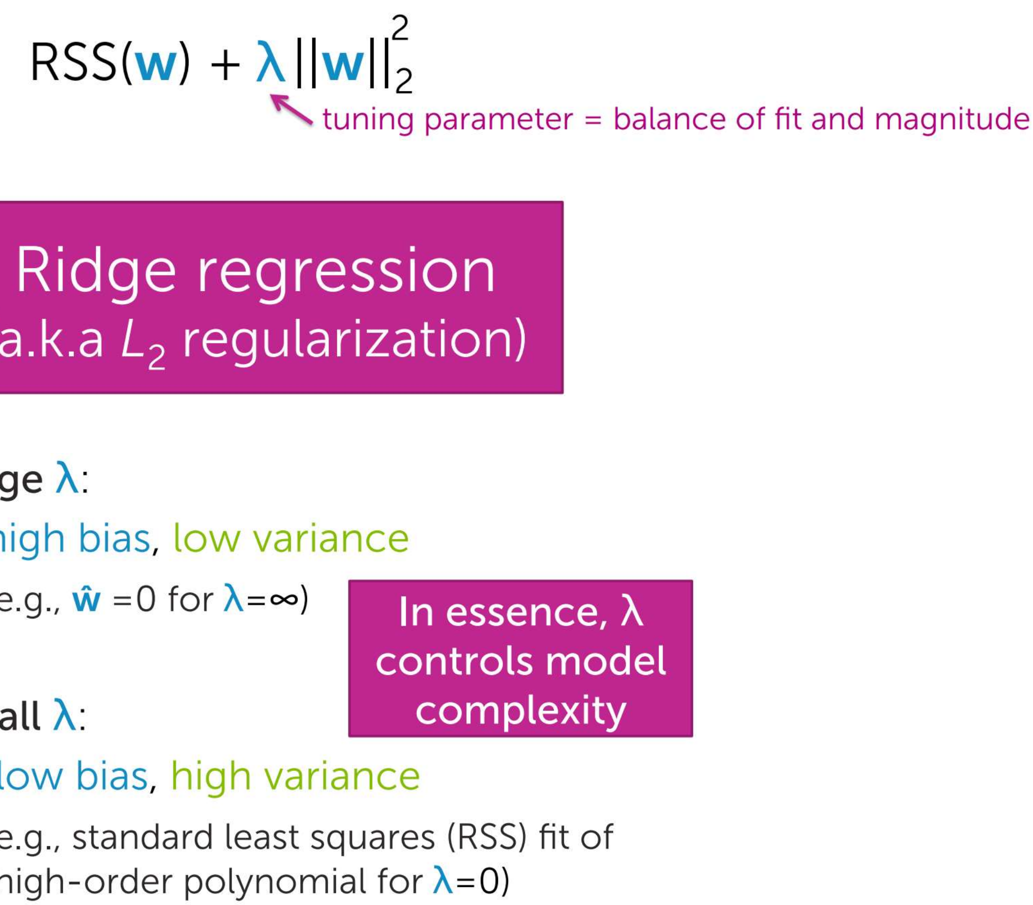


How to choose Lambda

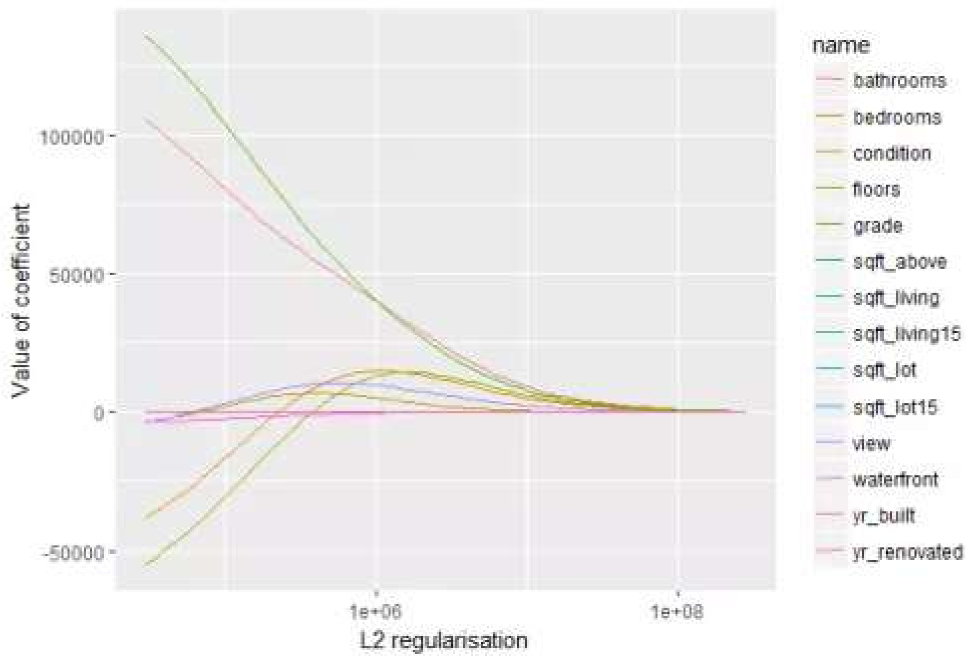


What is Ridge Regularisation

The L2 regularization adds a penalty equal to the sum of the squared value of the coefficients.



The L2 regularization will force the parameters to be relatively small, the bigger the penalization, the smaller (and the more robust) the coefficients are.



Difference between L1 and L2 regularization

L1 Regularization

- L1 penalizes sum of absolute value of weights.
- L1 has a sparse solution
- L1 has multiple solutions
- L1 has built in feature selection
- L1 is robust to outliers
- L1 generates model that are simple and interpretable but cannot learn complex patterns

L2 Regularization

- L2 regularization penalizes sum of square weights.
- L2 has a non sparse solution
- L2 has one solution

- L2 has no feature selection
- L2 is not robust to outliers
- L2 gives better prediction when output variable is a function of all input features
- L2 regularization is able to learn complex data patterns

Load the dataset

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectKBest, SelectPercentile
from sklearn.metrics import accuracy_score
```

In [3]:
```
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.feature_selection import SelectFromModel
```

In [4]:
```
data = pd.read_csv('0.9_5subjectslabelled_data.csv', nrows = 31437)
data.head()
```

Out[4]:

	Time Snap	AccX	AccY	AccZ	Gyro_X	Knee Angles	Gait Cycle Phase
0	120.775	-0.181472	-0.088708	-0.665352	0.087145	67.223821	5
1	120.780	-0.181443	-0.088745	-0.659870	0.103506	67.217858	5
2	120.785	-0.183826	-0.089735	-0.654215	0.117635	67.154903	5
3	120.790	-0.188545	-0.091706	-0.648504	0.129259	67.011479	5
4	120.795	-0.195535	-0.094645	-0.642857	0.138193	66.799616	5

In [5]:
```
X = data.drop('Gait Cycle Phase', axis = 1)
y = data['Gait Cycle Phase']

X.shape, y.shape
```

Out[5]:
```
((31436, 6), (31436,))
```

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
Time Snap            0
AccX                 0
AccY                 0
AccZ                 0
Gyro_X               0
Knee Angles          0
Gait Cycle Phase     0
dtype: int64
```

In [7]:
```
data.head()
```

Out[7]:

	Time Snap	AccX	AccY	AccZ	Gyro_X	Knee Angles	Gait Cycle Phase
0	120.775	-0.181472	-0.088708	-0.665352	0.087145	67.223821	5
1	120.780	-0.181443	-0.088745	-0.659870	0.103506	67.217858	5
2	120.785	-0.183826	-0.089735	-0.654215	0.117635	67.154903	5
3	120.790	-0.188545	-0.091706	-0.648504	0.129259	67.011479	5
4	120.795	-0.195535	-0.094645	-0.642857	0.138193	66.799616	5

In [8]:
```
X.shape, y.shape
```

Out[8]:
```
((31436, 6), (31436,))
```

In [9]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, ran
```

Estimation of coefficients of Linear Regression

In [10]:
```
sel = SelectFromModel(LinearRegression())
```

In [11]:
```
sel.fit(X_train, y_train)
```

Out[11]:
```
▸ SelectFromModel
▸ estimator: LinearRegression
  ▸ LinearRegression
```

In [12]:
```
sel.get_support()
```

Out[12]:
```
array([False,  True, False, False, False, False])
```

In [13]:
```
sel.estimator_.coef_
```

Out[13]:
```
array([ 0.00346151, -1.60165546, -0.03766687, -0.33100167, -0.02932753,
        0.08248934])
```

In [14]:
```
mean = np.mean(np.abs(sel.estimator_.coef_))
```

In [15]:
```
mean
```

Out[15]:
```
0.3476003973292832
```

In [16]:
```
np.abs(sel.estimator_.coef_)
```

Out[16]:
```
array([0.00346151, 1.60165546, 0.03766687, 0.33100167, 0.02932753,
       0.08248934])
```

In [17]:
```
features = X_train.columns[sel.get_support()]
features
```

Out[17]:
```
Index(['AccX'], dtype='object')
```

In [18]:
```
X_train_reg = sel.transform(X_train)
X_test_reg = sel.transform(X_test)
```

In [19]:
```
X_test_reg.shape
```

Out[19]:
```
(9431, 1)
```

In [20]:
```
def run_randomForest(X_train, X_test, y_train, y_test):
    clf = RandomForestClassifier(n_estimators=100, random_state=0, n_jobs=-1)
    clf = clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print('Accuracy: ', accuracy_score(y_test, y_pred))
```

In [21]:
```
%%time
run_randomForest(X_train_reg, X_test_reg, y_train, y_test)
```

```
Accuracy:  0.3426996076768105
CPU times: total: 26.3 s
Wall time: 5.76 s
```

In [22]:
```
%%time
run_randomForest(X_train, X_test, y_train, y_test)
```

```
Accuracy:  0.9212172622203372
CPU times: total: 29.5 s
Wall time: 7.07 s
```

In [23]:
```
X_train.shape
```

Out[23]:
```
(22005, 6)
```

Logistic Regression Coefficient with L1 Regularization

In [28]:
```
sel = SelectFromModel(LogisticRegression(penalty = 'l1', C = 0.001, solver = ''
sel.fit(X_train, y_train)
sel.get_support()
```

```
C:\Users\ibra5\AppData\Roaming\Python\Python38\site-packages\sklearn\svm\_bas
e.py:1244: ConvergenceWarning: Liblinear failed to converge, increase the num
ber of iterations.
  warnings.warn(
```

Out[28]:
```
array([ True,  True, False,  True,  True,  True])
```

In [29]:
```
sel.estimator_.coef_
```

Out[29]:
```
array([[ 0.00552602,  0.        ,  0.        , -0.21324871,  0.        ,
        -0.18081634],
       [-0.00270144,  0.        ,  0.        ,  0.        , -0.014516  ,
        -0.03474661],
       [ 0.00763851,  0.        ,  0.        ,  0.        ,  0.        ,
        -0.11690394],
       [ 0.0032008 ,  0.        ,  0.        ,  0.        ,  0.        ,
        -0.13527313],
       [-0.02126795, -0.13549613,  0.        ,  0.        ,  0.29649491,
         0.02958175],
       [-0.09962881,  0.        ,  0.        ,  0.        ,  0.        ,
         0.22140367],
       [-0.02226582,  0.        ,  0.        ,  0.        , -0.30866153,
         0.03732548]])
```

In [30]:
```
X_train_l1 = sel.transform(X_train)
X_test_l1 = sel.transform(X_test)
```

In [31]:
```
%%time
run_randomForest(X_train_l1, X_test_l1, y_train, y_test)
```

```
Accuracy:  0.894602905312268
CPU times: total: 27.8 s
Wall time: 6.18 s
```

L2 Regularization

In [37]:
```
sel = SelectFromModel(LogisticRegression(penalty = 'l2', C = 0.001, solver = 'l
sel.fit(X_train, y_train)
sel.get_support()
```

Out[37]:
```
array([False,  True, False,  True,  True, False])
```

In [38]:
```
sel.estimator_.coef_
```

Out[38]:
```
array([[ 0.00830478,  0.14460969, -0.16918171, -0.78213138, -0.04940001,
        -0.21145748],
       [-0.00167044,  0.0336125 ,  0.020244  ,  0.32903055, -0.14845027,
        -0.03613613],
       [ 0.01100884,  0.30676284,  0.13316299,  0.25528909, -0.00246275,
        -0.12211148],
       [ 0.00498958, -0.06504516, -0.00982899,  0.1279054 ,  0.14397183,
        -0.15511038],
       [-0.02811745, -0.77456044,  0.14922525,  0.04584105,  0.40598466,
         0.03736115],
       [-0.11344069,  0.22355685,  0.07274811,  0.12597473,  0.06330075,
         0.25656452],
       [-0.02924882, -0.52769859, -0.21616341, -0.46042748, -0.40879518,
         0.04441071]])
```

In [39]:
```
X_train_l2 = sel.transform(X_train)
X_test_l2 = sel.transform(X_test)
```

In [40]:
```
%%time
run_randomForest(X_train_l2, X_test_l2, y_train, y_test)
```

```
Accuracy:  0.7055455412999682
CPU times: total: 17.5 s
Wall time: 3.99 s
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: