Hilton Prediction Challenge

AGENDA

Background

BACKGROUND

How many analytics terms can you think of that are synonymous with "**HR Analytics**"?

HILTON PREDICTION CHALLENGE

BACKGROUND

HR Analytics

Workforce Analytics

People Analytics

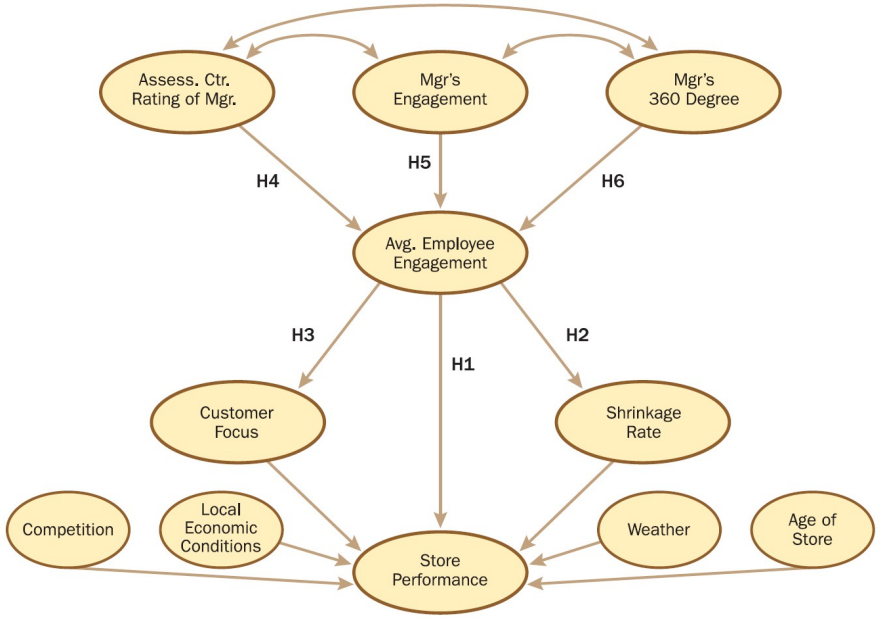
Talent Analytics

Human Capital Management

HILTON PREDICTION CHALLENGE

BACKGROUND

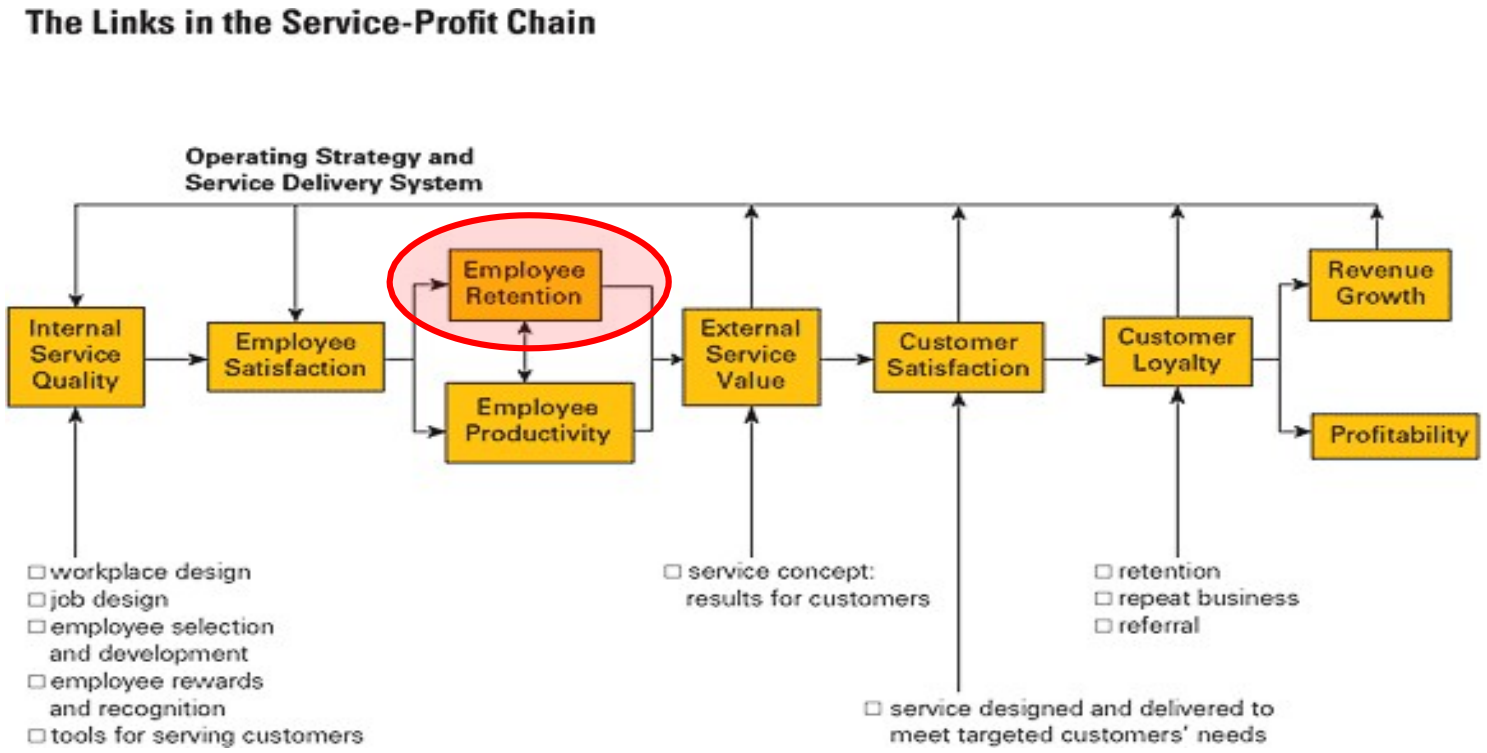
EXHIBIT 1: LOWE'S FIRST STORE MODEL BLUEPRINT



The initial Lowe's store model blueprint and roadmap for the final models represented initial hypotheses from the key stakeholders about how the data would interact in the model.

HILTON PREDICTION CHALLENGE

BACKGROUND



Code and Concepts

Background | **Code and Concepts** | Kaggle

HILTON PREDICTION CHALLENGE

CODE & CONCEPTS

Confusion Matrix

Feature Importance

Loading the Hilton Data (task, DV, and variables)

Data Splitting

Decision Tree Visual

Grid Search

XGBoost

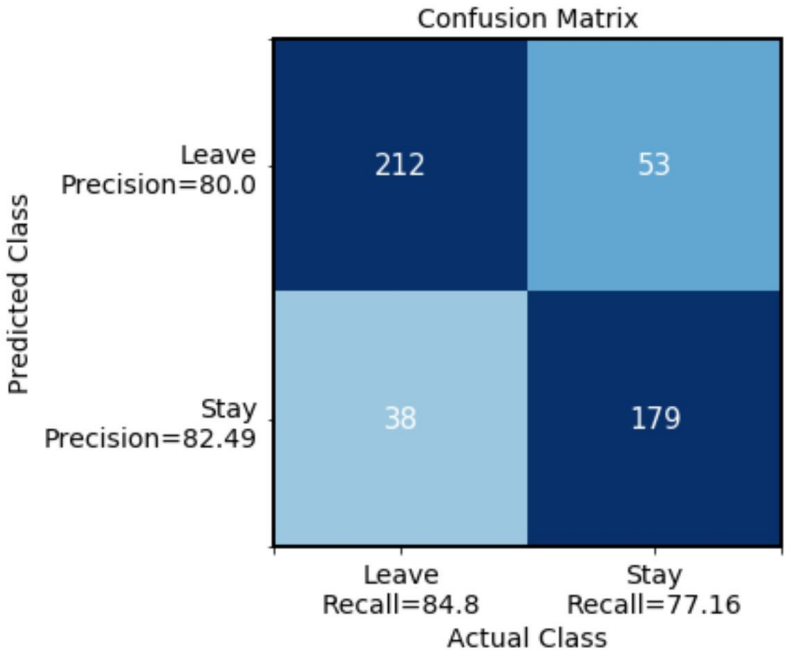
ROC Curve

ROC Variance

HILTON PREDICTION CHALLENGE

CODE & CONCEPTS – CONFUSION MATRIX

```
def displayConfusionMatrix(confusionMatrix, precisionNegative, precisionPositive, recallNegative, recallPositive, ti
    # Set font size for the plots. You can ignore this line.
    PLOT_FONT_SIZE = 14

    # Set plot size. Please ignore this line
    plt.rcParams['figure.figsize'] = [5, 5]

    # Transpose of confusion matrix to align the plot with the actual precision recall values. Please ignore this as
    confusionMatrix = np.transpose(confusionMatrix)

    # Plotting the confusion matrix
    plt.imshow(confusionMatrix, interpolation='nearest',cmap=plt.cm.Blues, vmin=0, vmax=100)


    # Setting plot properties. You should ignore everything from here on.
    xticks = np.array([-0.5, 0, 1,1.5])
    plt.gca().set_xticks(xticks)
    plt.gca().set_yticks(xticks)
    plt.gca().set_xticklabels(["", "Leave\nRecall=" + str(recallNegative) , "Stay\nRecall=" + str(recallPositive), "
    plt.gca().set_yticklabels(["", "Leave\nPrecision=" + str(precisionNegative) , "Stay\nPrecision=" + str(precisior
    plt.ylabel("Predicted Class", fontsize=PLOT_FONT_SIZE)
    plt.xlabel("Actual Class", fontsize=PLOT_FONT_SIZE)
    plt.title(title, fontsize=PLOT_FONT_SIZE)

    # Add text in heatmap boxes
    for i in range(2):
        for j in range(2):
            text = plt.text(j, i, confusionMatrix[i][j], ha="center", va="center", color="white", size=15) ### size

    plt.show()
```



HILTON PREDICTION CHALLENGE

CODE & CONCEPTS – LOADING THE HILTON DATA

4.1 Data Loading with Pandas

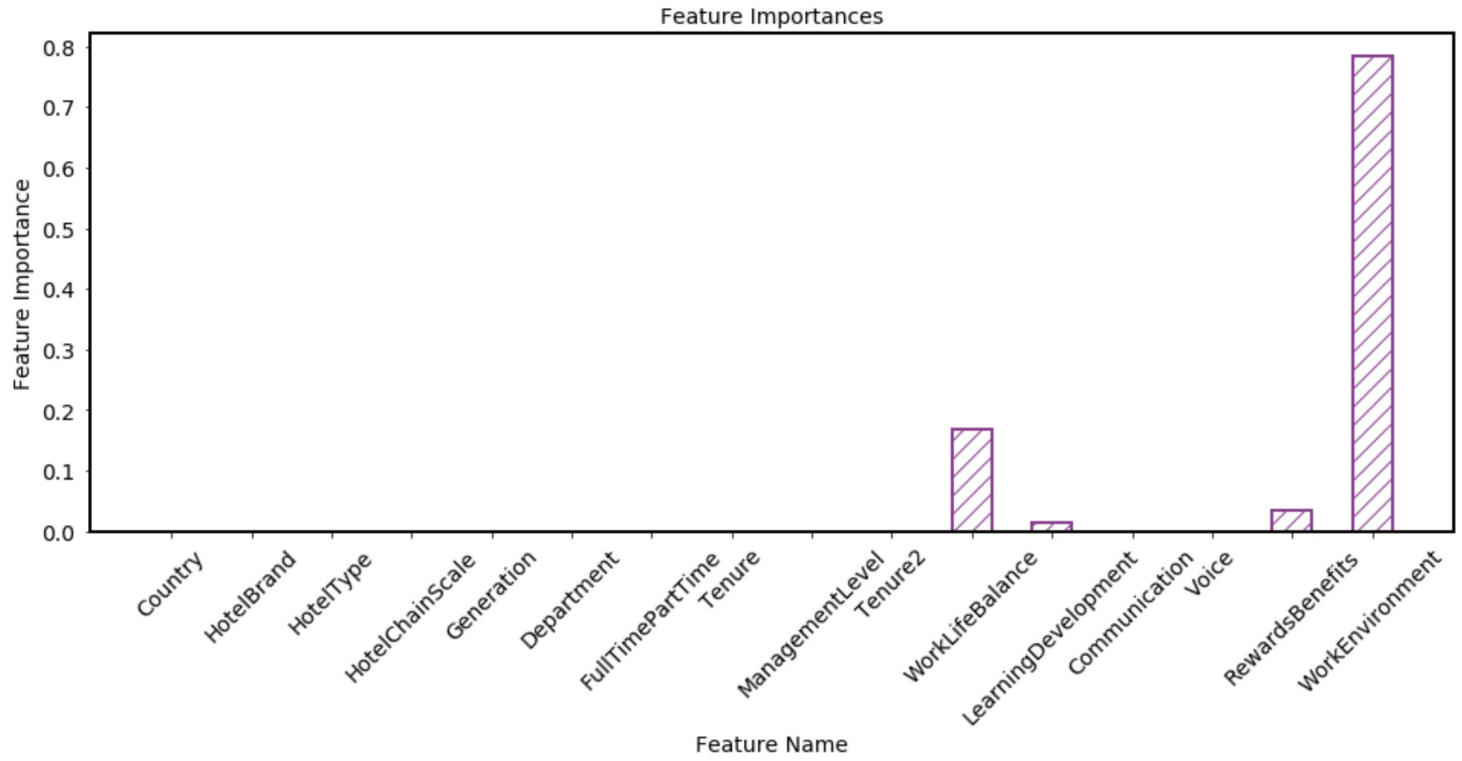
[7]:
```
# Read data into a data frame
data = pd.DataFrame(pd.read_csv("Data/HiltonPredictionData_Train.csv", ","))

# Delete null values
data = data.dropna()

# Delete columns for dependent variables
data = data.drop(columns=['Engagement', 'JobSatisfaction', 'PaySatisfaction', 'RecommendToWork', 'RecommendFriendsToStay'])

# Get column names. We will use these for visualization purposes
columns = list(data.columns)

# Display the data frame as a table
display(data)
```

	AnonymousID	HotelInncode	Country	HotelBrand	HotelType	HotelChainScale	Generation	Department	FullTimePartTime	Tenure	Manageme
0	2	100038	1	4	2	3	2	20	1	4	
1	3	100038	1	4	2	3	1	16	1	2	
2	4	100038	1	4	2	3	1	16	1	1	
3	5	100038	1	4	2	3	1	16	1	2	
4	7	100038	1	4	2	3	2	20	1	4	

HILTON PREDICTION CHALLENGE

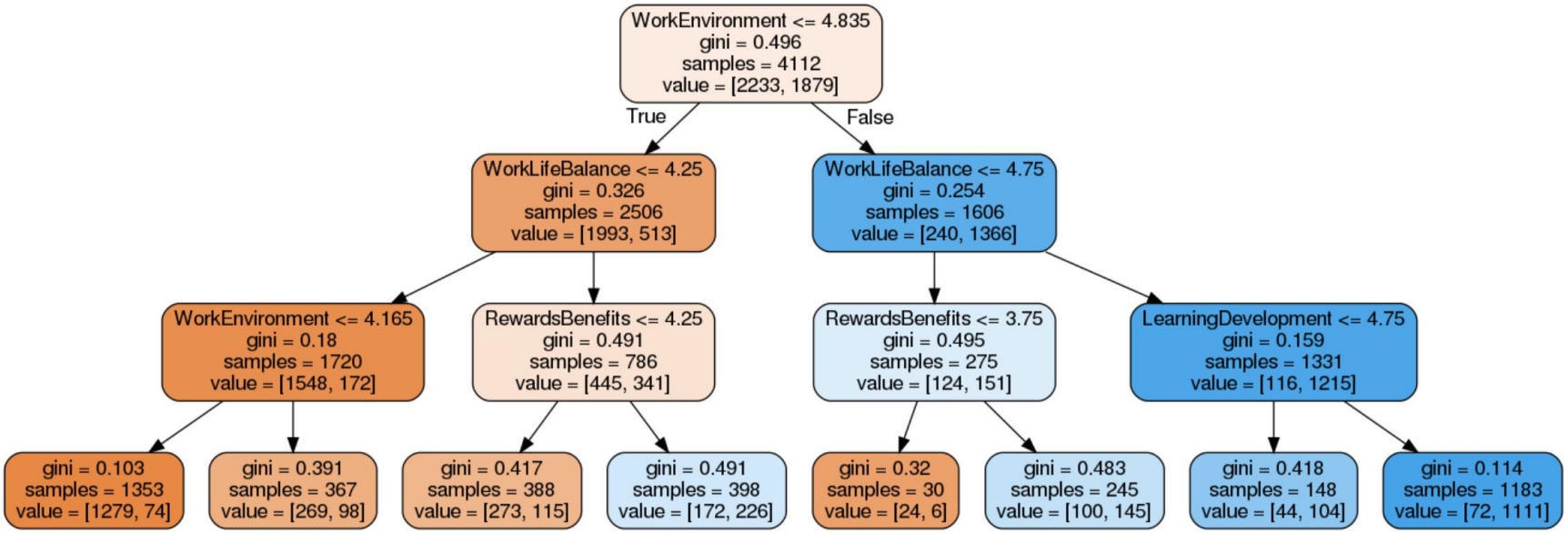
CODE & CONCEPTS – FEATURE IMPORTANCE

```
# Calculate feature importance
showFeatureImportance(classifierDt, columns)
```



HILTON PREDICTION CHALLENGE

CODE & CONCEPTS – DECISION TREE VISUAL



HILTON PREDICTION CHALLENGE

CODE & CONCEPTS – DATA SPLIT

4.3 Data Splitting

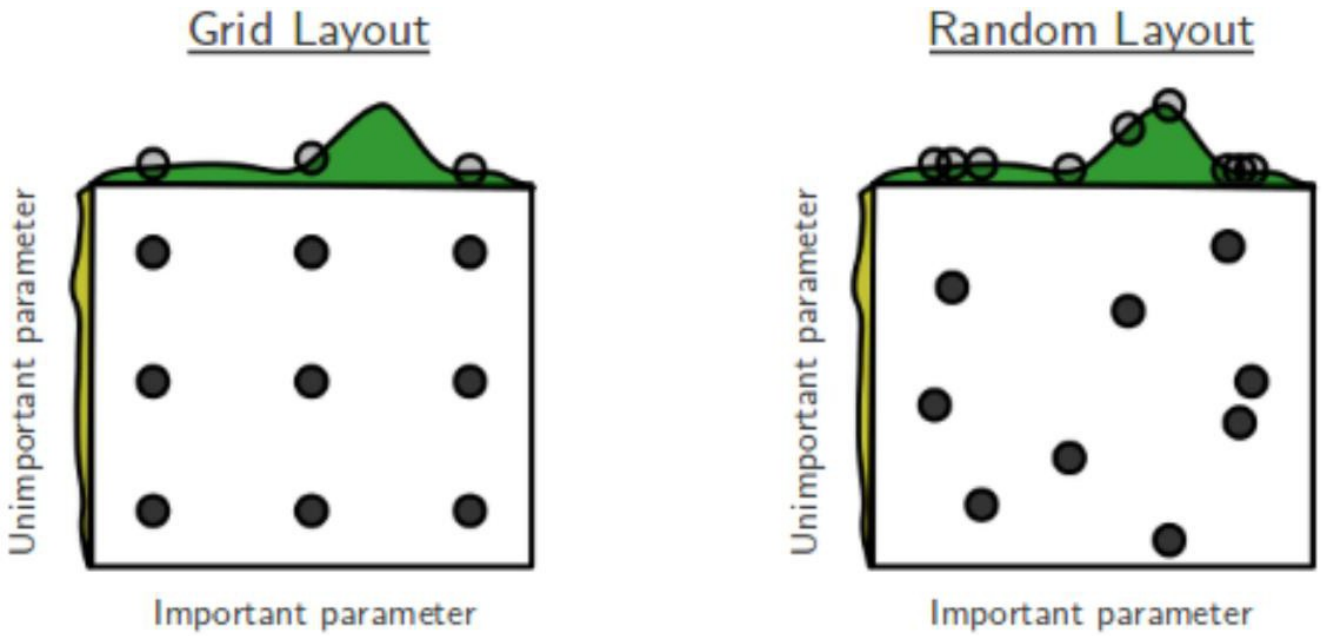
[32]:
```
# Test data percentage (0.1 = 10%)
TEST_DATA_PERCENTAGE = 0.1

# Validation data percentage
VALIDATION_DATA_PERCENTAGE = 0.05

# Split into train and test
trainData, testData, trainLabels, testLabels = train_test_split(features, labels, test_size=TEST_DATA_PERCENTAGE)
```

Training Data
(used to build the model)

Validation Data
(this is called test data in the notebook)

Test Data
(this is the Kaggle dataset: Section 5.6)

CODE & CONCEPTS – GRID SEARCH



HILTON PREDICTION CHALLENGE

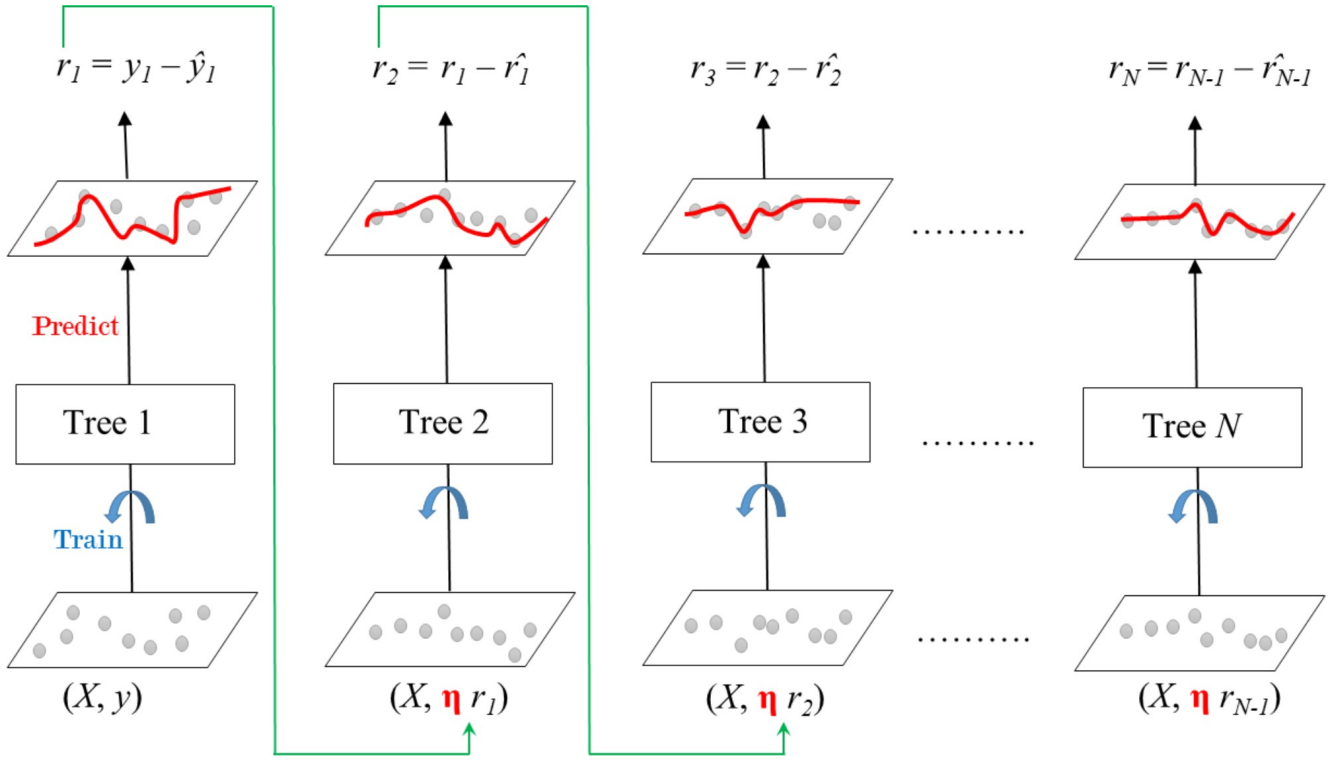
CODE & CONCEPTS – GRID SEARCH

```
##########################################################################################################################
# GridSearchCV function takes in a parameter dictionary which is very important here.
# The dictionary contains as keys the parameter names and values are the possible values of the parameter that you w
parameters = {'max_depth': [2, 4, 6]} # You can add different depths here.
##########################################################################################################################
```

[40]:

```
###### We will repeat the same process we did in the previous DecisionTreeClassifier cell but we'll now wrap our cla
# Create an instance of the decision tree algorithm. You can consider "classifier" as a function that is going to ge
classifierDtGrid = GridSearchCV(DecisionTreeClassifier(), parameters)

# Train the decision tree model using the function .fit
classifierDtGrid.fit(trainData, trainLabels)

# Calculate training accuracy of the classifier
trainAccuracy = classifierDtGrid.score(trainData, trainLabels)

# Predict on test data
predictions = classifierDtGrid.predict(testData) # This will give binary labels e.g 0/1
predictionProbabilities = classifierDtGrid.predict_proba(testData) # This will give prediction probabilities for bot
```

CODE & CONCEPTS – XGBOOST



CODE & CONCEPTS – XGBOOST

5.3.1 Grid Search on XgBoost

XgBoost has so many parameters that you can tune. Let us try to do a grid search on those parameters to get the best model possible. The list of all parameters is listed here:

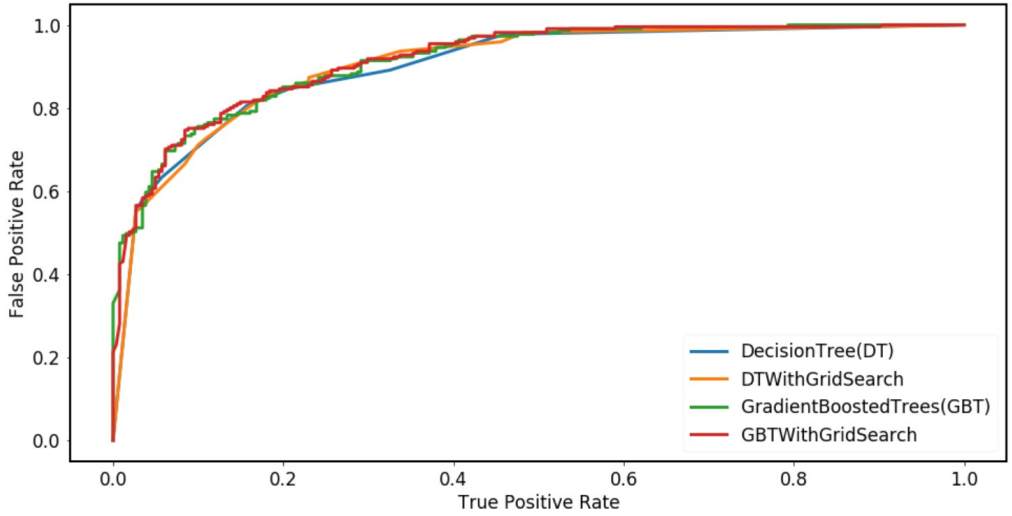
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

[46]:
```
set_background('#ffffff')
##########################################################################################################################
# Things you can change
##########################################################################################################################
# GridSearchCV function takes in a parameter dictionary which is very important here.
# The dictionary contains as keys the parameter names and values are the possible values of the parameter that you w
parameters = {'max_depth': [6, 10],
              'learning_rate': [0.1,0.2],
             'n_estimators': [20, 100],
             'subsample': [1.0]} # You can add different depths here.

# In order to learn about these parameters and their possible values, please go to:
# https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn.ensembl
##########################################################################################################################
```

CODE & CONCEPTS – ROC CURVE

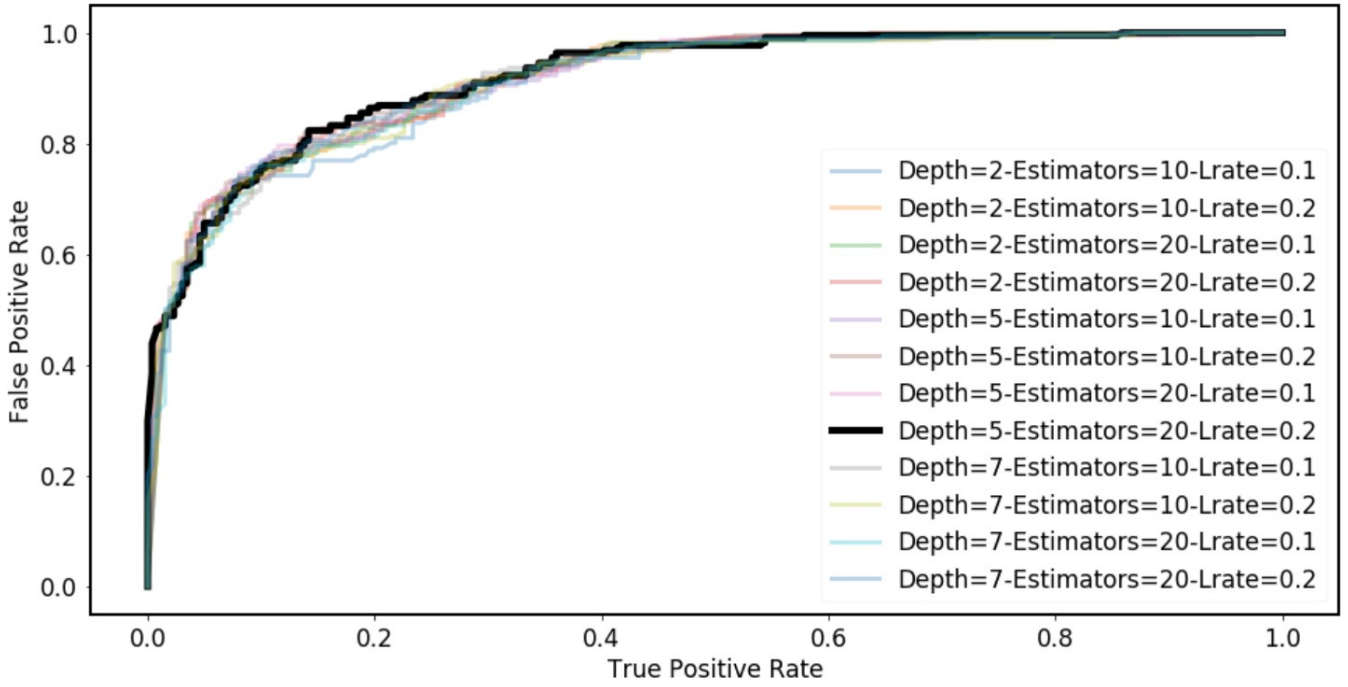
5.4 ROC Curve

[49]:
```
plt.rcParams['figure.figsize'] = [16, 8]
FONT_SIZE = 17

plt.plot(falsePositiveRateDt, truePositiveRateDt, linewidth = 3, label = "DecisionTree(DT)")
plt.plot(falsePositiveRateDtWithGridSearch, truePositiveRateDtWithGridSearch, linewidth = 3, label = "DTWithGridSearch")
plt.plot(falsePositiveRateGb, truePositiveRateGb, linewidth = 3, label = "GradientBoostedTrees(GBT)")
plt.plot(falsePositiveRateGbWithGridSearch, truePositiveRateGbWithGridSearch, linewidth = 3, label = "GBTWithGridSearch")

plt.legend(fontsize=FONT_SIZE)
plt.xticks(fontsize=FONT_SIZE)
plt.yticks(fontsize=FONT_SIZE)
plt.xlabel("True Positive Rate", fontsize=FONT_SIZE)
plt.ylabel("False Positive Rate", fontsize=FONT_SIZE)
plt.show()
```



HILTON PREDICTION CHALLENGE

CODE & CONCEPTS – ROC VARIANCE



HILTON PREDICTION CHALLENGE

SUBMISSION

```
resultsFile = open("Results/dtPredictions.csv", "w")
resultsFile.write("Id,Prediction\n")
for predictionProb, i in zip(classifierDt.predict_proba(testFeaturesForKaggle), np.arange(len(testFeaturesForKaggle))):
    resultsFile.write(str(i + 1) + "," + str(float(predictionProb[1])) + "\n")
resultsFile.close()

resultsFile = open("Results/dtPredictionsGrid.csv", "w")
resultsFile.write("Id,Prediction\n")
for predictionProb, i in zip(classifierDtGrid.predict_proba(testFeaturesForKaggle), np.arange(len(testFeaturesForKaggle
    resultsFile.write(str(i + 1) + "," + str(float(predictionProb[1])) + "\n")
resultsFile.close()

resultsFile = open("Results/gbPredictions.csv", "w")
resultsFile.write("Id,Prediction\n")
for predictionProb, i in zip(gbClassifier.predict_proba(testFeaturesForKaggle), np.arange(len(testFeaturesForKaggle))):
    resultsFile.write(str(i + 1) + "," + str(float(predictionProb[1])) + "\n")
resultsFile.close()

resultsFile = open("Results/gbPredictionsGrid.csv", "w")
resultsFile.write("Id,Prediction\n")
for predictionProb, i in zip(gbClassifierWithGridSearch.predict_proba(testFeaturesForKaggle), np.arange(len(testFeature
    resultsFile.write(str(i + 1) + "," + str(float(predictionProb[1])) + "\n")
resultsFile.close()

print("All predictions have been placed in the results folder...")
```

?


VIRGINIA