Mental Health Treatment Prediction

Group 6

Richard
Desmond
Darren
Kurtis

Content

1. Goals of Our Research (Business Question of Our Project)
2. Preprocessing
3. Model Selection
4. Model Evaluation
5. Conclusion and Future Improvement

01

Background and Goal

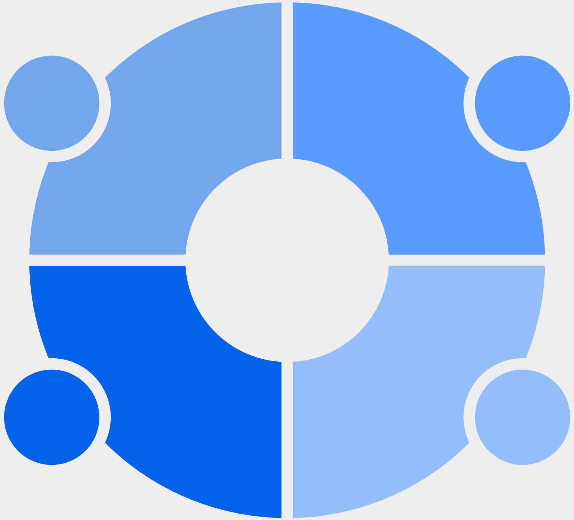
Goal

To see if we can use machine learning to predict whether or not the participant attended treatment

Business Values for Companies

Identify the needs for mental health resources

Re-evaluate the health insurance coverage



Satisfaction assessment criteria among Employees

Develop a more effective diagnosis protocol

Dataset:
Mental Health in Tech Survey

Dataset from a 2014 survey that measures respondents' attitudes towards mental health and frequency of mental health disorders in the tech workplace.

27 variables/columns included (but not limited to):

- Respondent's age, gender, country
- Respondents/families treatment history
- Respondent's view towards mental health support in their workplace

Assumptions & Hypothesis

Age

18-54, average age at onset is the mid-20s

Family History

2-6x risk than those without

Male & LGBTQ

4-6x higher in mental disorders & suicidal rate

Resource Availability

Lack of mental health support

Self-employed

Higher risk of having mental illness

References:
https://www.hopkinsmedicine.org/health/wellness-and-prevention/mental-health-disorder-statistics
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6761480/
https://www.cdc.gov/workplacehealthpromotion/tools-resources/pdfs/issue-brief-no-2-mental-health-and-chronic-disease.pdf

02

EDA & Preprocessing

Outlier Handling

```
# EDA - Check unique values of age
# Because there are outliers in age
df.Age.unique()
```

```
# EDA - Create a df of normal age first
normalage = df.query('Age < 100 & Age > 0')
normalage['Age'].unique()
```

```
array([37, 44, 32, 31, 33, 35, 39, 42, 23, 29, 36, 27, 46, 41, 34, 30, 40,
       38, 50, 24, 18, 28, 26, 22, 19, 25, 45, 21, 43, 56, 60, 54, 55, 48,
       20, 57, 58, 47, 62, 51, 65, 49,  5, 53, 61,  8, 11, 72],
      dtype=int64)
```

Data Cleaning

```
# Regroup Gender
df[df["Gender"] == "Mail"] = df[df["Gender"] == "Mail"].replace("Mail", "Male")
df[df["Gender"] == "Malr"] = df[df["Gender"] == "Malr"].replace("Malr", "Male")
df[df["Gender"] == "Cis Man"] = df[df["Gender"] == "Cis Man"].replace("Cis Man", "Male")
df[df["Gender"] == "male"] = df[df["Gender"] == "male"].replace("male", "Male")
df[df["Gender"] == "M"] = df[df["Gender"] == "M"].replace("M", "Male")
df[df["Gender"] == "m"] = df[df["Gender"] == "m"].replace("m", "Male")
df[df["Gender"] == "Make"] = df[df["Gender"] == "Make"].replace("Make", "Male")
df[df["Gender"] == "Male "] = df[df["Gender"] == "Male "].replace("Male ", "Male")
df[df["Gender"] == "Cis Male"] = df[df["Gender"] == "Cis Male"].replace("Cis Male", "Male")
df[df["Gender"] == "Man"] = df[df["Gender"] == "Man"].replace("Man", "Male")
df[df["Gender"] == "maile"] = df[df["Gender"] == "maile"].replace("maile", "Male")
df[df["Gender"] == "Mal"] = df[df["Gender"] == "Mal"].replace("Mal", "Male")
df[df["Gender"] == "msle"] = df[df["Gender"] == "msle"].replace("msle", "Male")
df[df["Gender"] == "Male (CIS)"] = df[df["Gender"] == "Male (CIS)"].replace("Male (CIS)", "Male")
df[df["Gender"] == "cis male"] = df[df["Gender"] == "cis male"].replace("cis male", "Male")
df[df["Gender"] == "female"] = df[df["Gender"] == "female"].replace("female", "Female")
df[df["Gender"] == "F"] = df[df["Gender"] == "F"].replace("F", "Female")
df[df["Gender"] == "f"] = df[df["Gender"] == "f"].replace("f", "Female")
df[df["Gender"] == "Woman"] = df[df["Gender"] == "Woman"].replace("Woman", "Female")
df[df["Gender"] == "Female "] = df[df["Gender"] == "Female "].replace("Female ", "Female")
df[df["Gender"] == "Femake"] = df[df["Gender"] == "Femake"].replace("Femake", "Female")
```

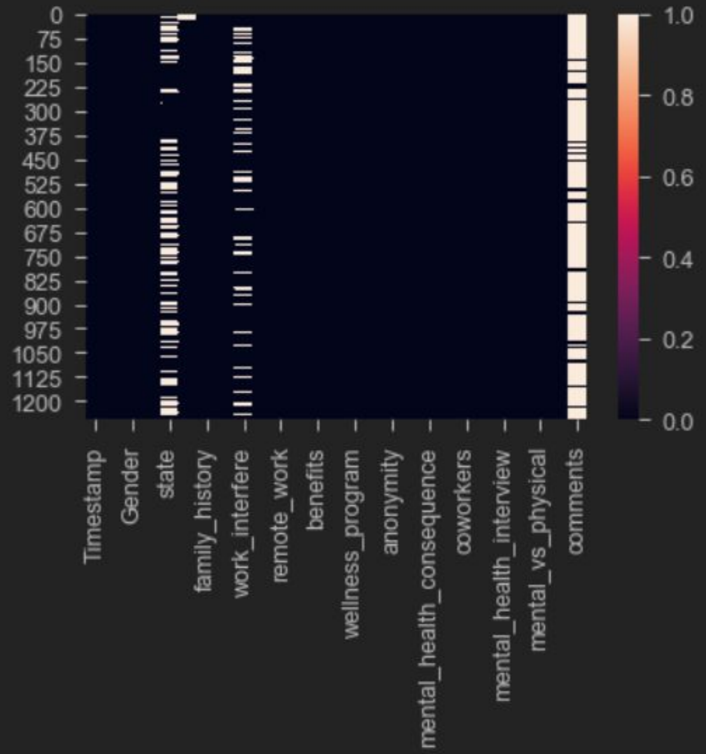
```
# Regroup Gender
df.loc[(df['Gender']!='Male') & (df['Gender']!='Female'),'Gender'] = 'Others'
```

```
# Create new columns from countries to continents: US & Non-US
df['US_or_not'] = np.where(df.Country == 'United States', 'US', 'Non_US')
```

Missing Values

```
# EDA - Check NA values
sns.heatmap(df.isnull())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c191f92088>
```



03

Model Selection

Selecting the Target Feature and Features

Target Feature

- Treatment
    - "Have you sought treatment for a mental health condition?"

Features

- Age, gender, country
- Family history
- View towards mental health support in their workplace

Modeling Classification

01

kNN

02

Logistic Regression

03

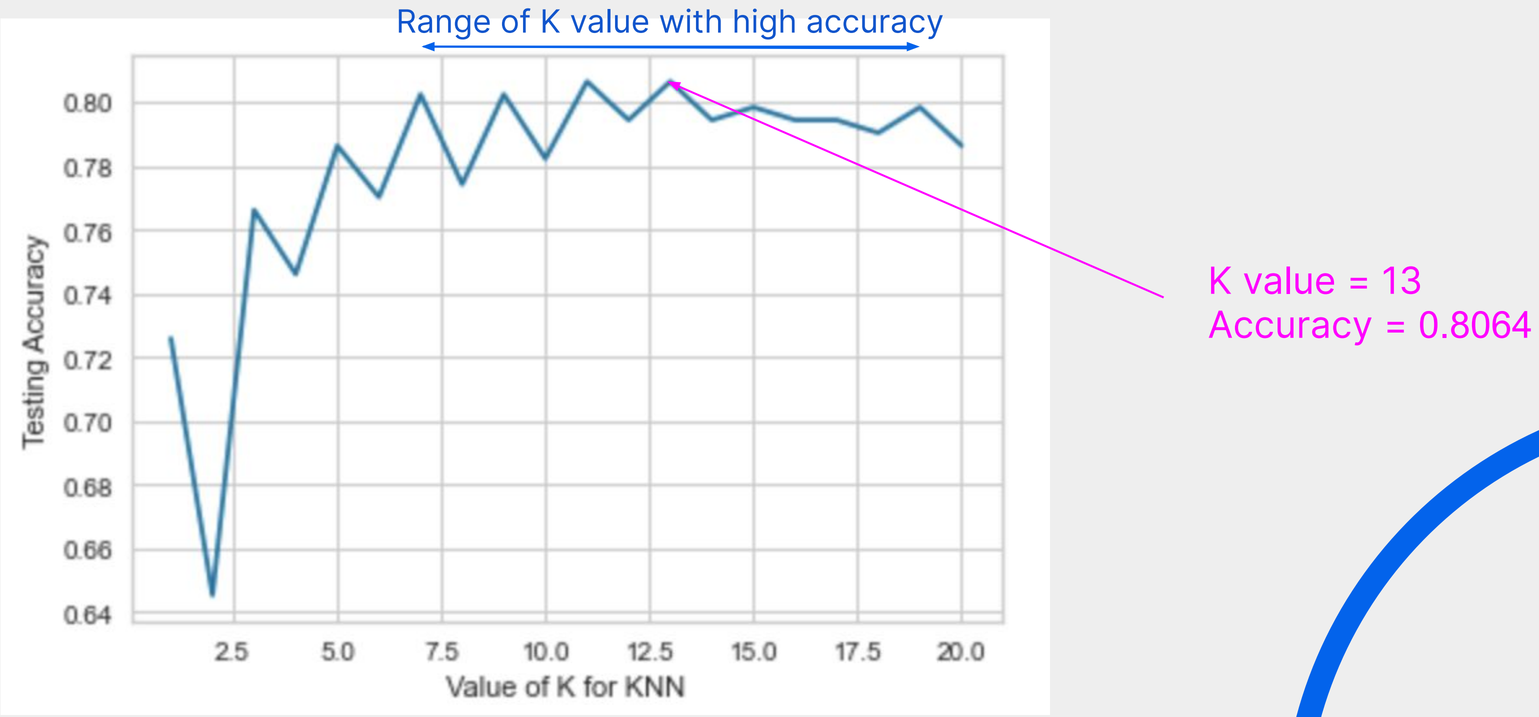
Random Forest

04

XGBoost

01 kNN

Creating model with KNeighborsClassifier in a range of 1 to 20



01 kNN

Hyperparameter Tuning

GridSearchCV: grid = GridSearchCV(knn, param_grid=params, cv=10)
Best_params = {'n_neighbors': 15}
Best_score = 0.8128942486085343

Cross-validation: knn = KNeighborsClassifier(n_neighbors=15) & cv=10
scores.mean() = 0.8128942486085343

Accuracy ↑ 1% after tuning

02 Logistic Regression

```
from sklearn.model_selection import GridSearchCV
grid={"C":np.logspace(-3,3,7), "penalty":["l1","l2"]}
logreg_cv=GridSearchCV(logmodel,grid,cv=10)
logreg_cv.fit(X_train,y_train)
```

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2'}
accuracy : 0.8270253555967843
```

03 Random Forest

Model Building

- Build the RF with default parameters

Hyperparameter Tuning

- RandomizedSearchCV to obtain performing parameters in brief, i.e. n_estimators, max_depth, max_features
- GridSearchCV to refine and narrow down for best parameters, i.e. min_samples_split, min_samples_leaf, criterion, bootstrap + above params

Test Report:

	precision	recall	f1-score	support
0	0.80	0.73	0.76	117
1	0.77	0.84	0.81	131
accuracy			0.79	248
macro avg	0.79	0.78	0.78	248
weighted avg	0.79	0.79	0.79	248

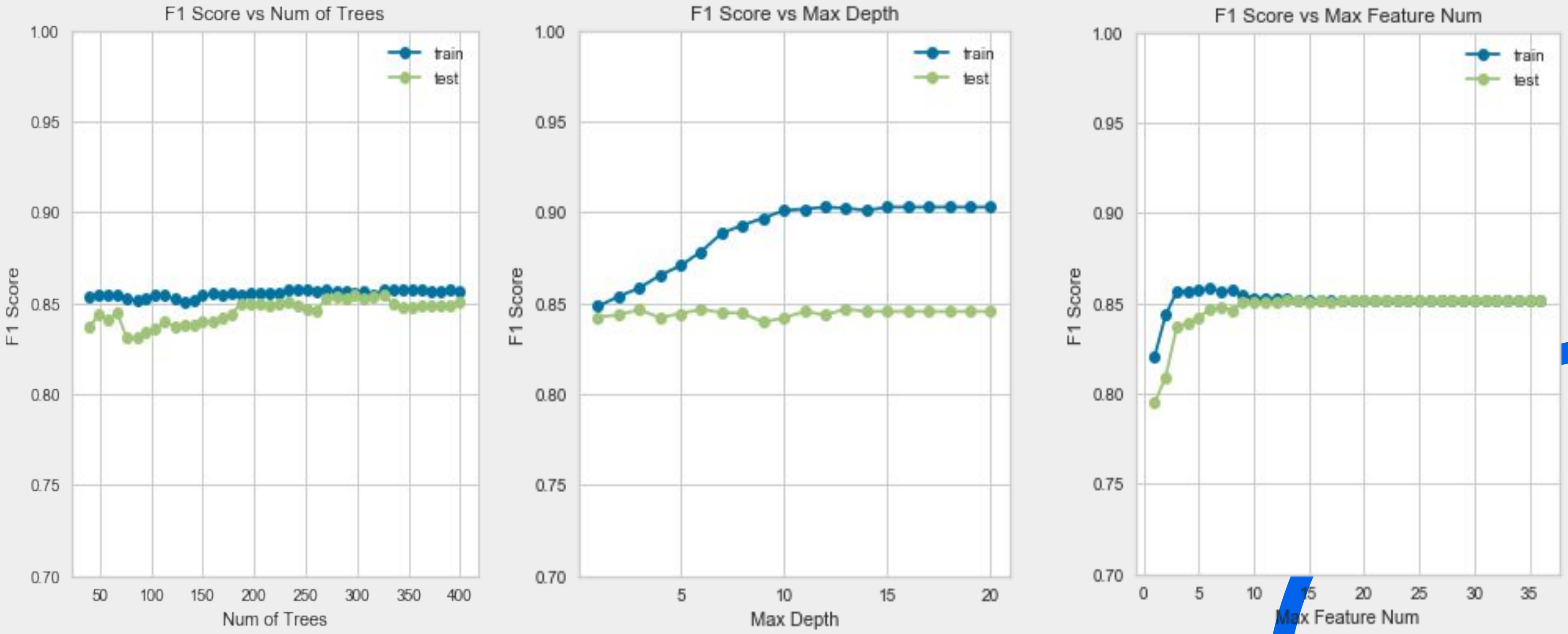
3% higher

Tuned Report:

	precision	recall	f1-score	support
0	0.88	0.70	0.78	117
1	0.77	0.92	0.84	131
accuracy			0.81	248
macro avg	0.83	0.81	0.81	248
weighted avg	0.82	0.81	0.81	248

03 Random Forest

Optimal points of three most paramount parameters - Num of trees, Max Depth, Max Features



04 XGBoost

Model Building- with Default Parameters

```
#XGBoosting:Fit model into training data
from xgboost import XGBClassifier
model = XGBClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

	precision	recall	f1-score	support
0	0.76	0.75	0.76	117
1	0.78	0.79	0.78	131
accuracy			0.77	248
macro avg	0.77	0.77	0.77	248
weighted avg	0.77	0.77	0.77	248

04 XGBoost

Hyperparameter Tuning

```
#XGBoost Hyperparameter tunning-GridSearch CV
from sklearn.model_selection import GridSearchCV

model = XGBClassifier(random_state = 42, max_depth= 3, colsample_bytree = 0.06)
param = {'min_child_weight':[1,2,3,4], 'n_estimators':(180,1000)  ,'learning_rate':[0.13,0.14,0.015,0.16,0.1
grid = GridSearchCV(model, param_grid= param, cv=7)
grid.fit(X_train, y_train)
print(grid.best_score_)
print(grid.best_params_)
```

```
0.8279963468755797
{'learning_rate': 0.16, 'min_child_weight': 1, 'n_estimators': 180}
```

04 XGBoost

Hyperparameter Tuning

	precision	recall	f1-score	support
0	0.76	0.75	0.76	117
1	0.78	0.79	0.78	131
accuracy			0.77	248
macro avg	0.77	0.77	0.77	248
weighted avg	0.77	0.77	0.77	248

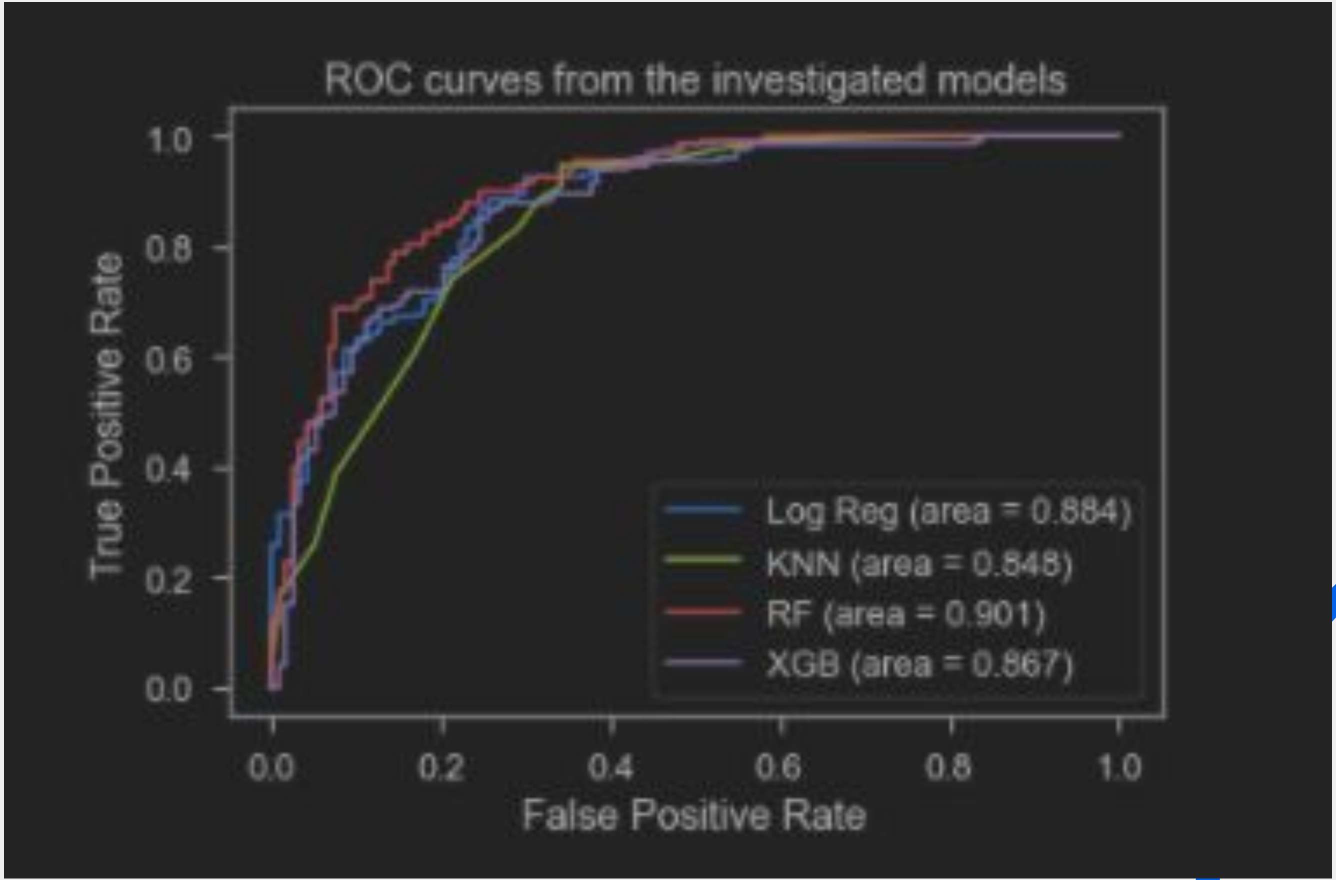
5% higher

	precision	recall	f1-score	support
0	0.86	0.74	0.79	117
1	0.79	0.89	0.84	131
accuracy			0.82	248
macro avg	0.83	0.81	0.82	248
weighted avg	0.82	0.82	0.82	248

04

Model Evaluation

Model Comparison: with ROC Curves



Model Comparison: with AUC and F1 Score

	AUC	Average F1 Score (After Tuning)
kNN	0.848	0.79
Logistic Regression	0.884	0.82
Random Forest	0.901	0.81
XGBoost	0.867	0.81

Strongest Predictor -
Whether Participant Joined Mental Health Treatment

	Assumptions	Feature Importance
Age	18-54	25-35
Family History	Yes	Yes
Gender	Male & LGBTQ	Male
Resource Availability	Little/No	Provided
Employment Mode	Self-employed	Insignificant

Conclusion and Future Improvement

- Based on the responses, we are able to predict whether the respondent have sought for medical treatment for mental health fairly accurately
- Difficult to predict whether respondents have needs for mental health support, due to the setting of the questionnaire and the privacy issues come along
- In the future :
  - Further analysis on comments using NLP technique
  - Test with more boosting models which might yield a more accurate model
  - Try unsupervised model like t-SNE for cluster identification
  - Retain states for more geographical analysis

The End

Appendix - Feature Importance