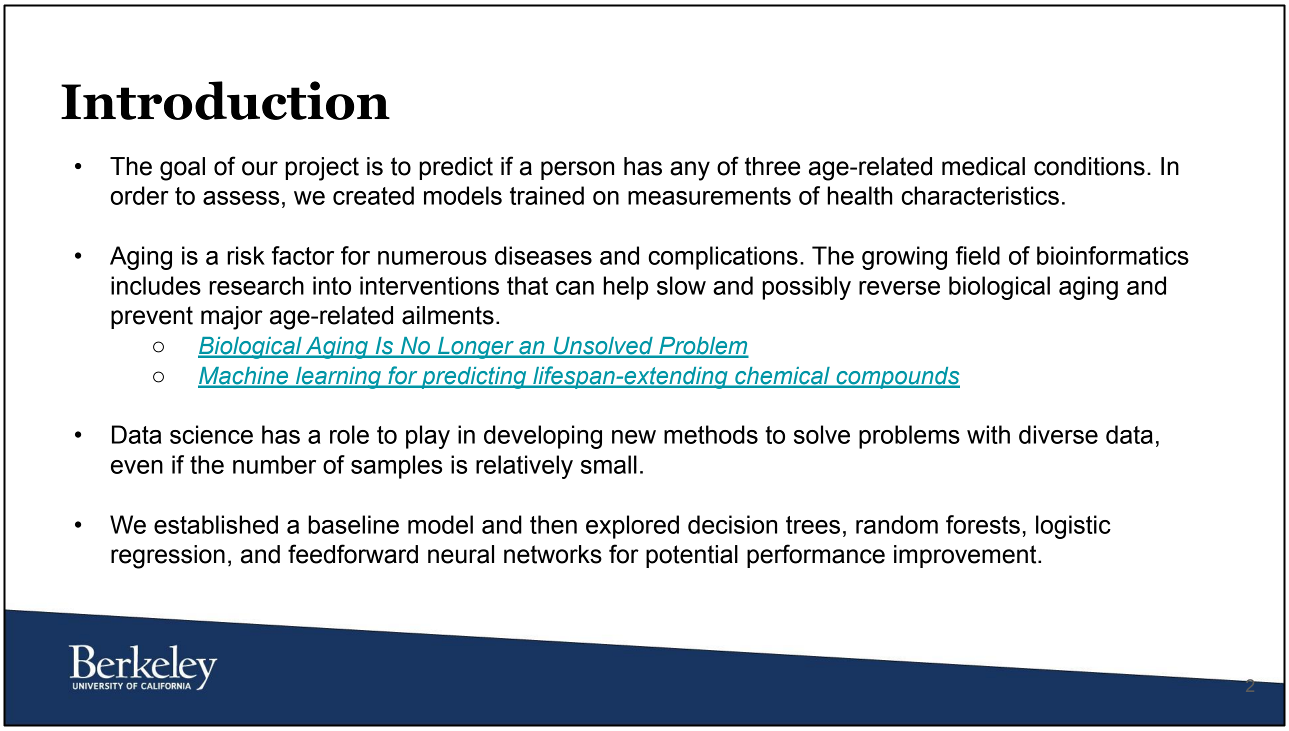
Identifying Age-Related Conditions

August 9, 2023

Akhila Ganti
Megan Nguyen
Leslie Nie
Kevin Stallone
Tim Tung

Berkeley
UNIVERSITY OF CALIFORNIA



Introduction

- The goal of our project is to predict if a person has any of three age-related medical conditions. In order to assess, we created models trained on measurements of health characteristics.
- Aging is a risk factor for numerous diseases and complications. The growing field of bioinformatics includes research into interventions that can help slow and possibly reverse biological aging and prevent major age-related ailments.
  - *Biological Aging Is No Longer an Unsolved Problem*
  - *Machine learning for predicting lifespan-extending chemical compounds*
- Data science has a role to play in developing new methods to solve problems with diverse data, even if the number of samples is relatively small.
- We established a baseline model and then explored decision trees, random forests, logistic regression, and feedforward neural networks for potential performance improvement.

Berkeley
UNIVERSITY OF CALIFORNIA

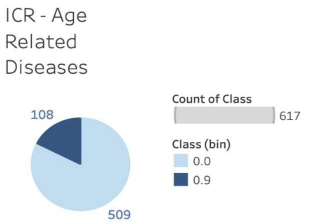
Akhila

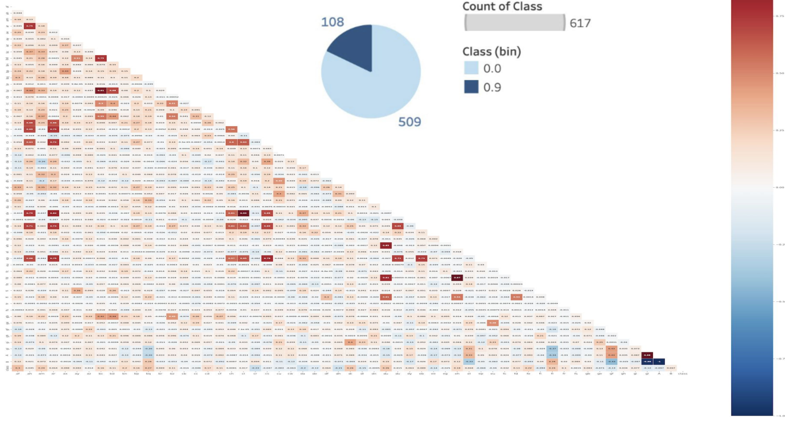
- What/Why/How
- We will need to predict if the person has one or more of any of the three medical conditions (Class 1), or none of the three medical conditions (Class 0).
- EDA part 1 and part 2

Now Kevin will discuss EDA

EDA – Initial Observations

- Data is sourced from an active (June 2023) Kaggle competition
- Small dataset (617 observations)
- Imbalanced target class:
  - 82.5% of class 0
  - 17.5% of class 1
- 56 health characteristic features
- Possibly multicollinearity from the correlation matrix
- Feature selection will be considered



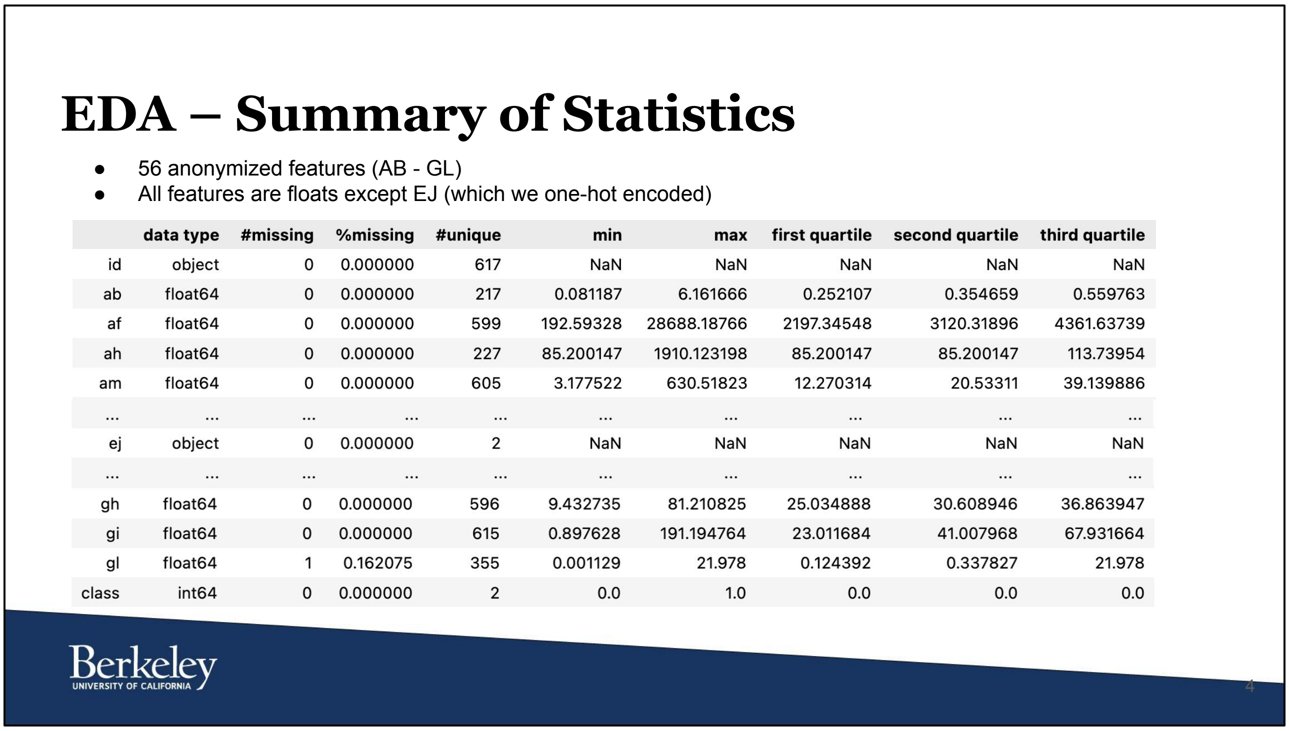


Berkeley
UNIVERSITY OF CALIFORNIA

Our data came from the month of June's Kaggle competition hosted by InVitro Cell Research, LLC (ICR). The competition's goal is to help identify Age-Related Conditions. That is whether someone has a diagnosis with an age related condition or not.

The dataset provided by them was small and contained only 617 observations with a target class imbalance of roughly 80% with no diagnosis to 20% with a diagnosis. As for features, It contained 56 health characteristics.

We also noticed a possible multicollinearity from the correlation heatmap matrix presented here so we placed an importance on our feature selection



EDA – Summary of Statistics

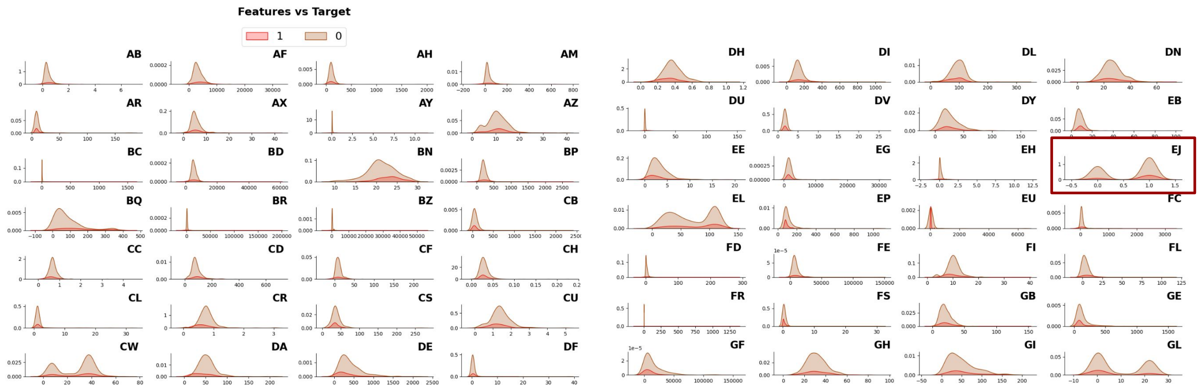
- 56 anonymized features (AB - GL)
- All features are floats except EJ (which we one-hot encoded)

	data type	#missing	%missing	#unique	min	max	first quartile	second quartile	third quartile
id	object	0	0.000000	617	NaN	NaN	NaN	NaN	NaN
ab	float64	0	0.000000	217	0.081187	6.161666	0.252107	0.354659	0.559763
af	float64	0	0.000000	599	192.59328	28688.18766	2197.34548	3120.31896	4361.63739
ah	float64	0	0.000000	227	85.200147	1910.123198	85.200147	85.200147	113.73954
am	float64	0	0.000000	605	3.177522	630.51823	12.270314	20.53311	39.139886
...	...	...	...	...	...	...	...	...	...
ej	object	0	0.000000	2	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
gh	float64	0	0.000000	596	9.432735	81.210825	25.034888	30.608946	36.863947
gi	float64	0	0.000000	615	0.897628	191.194764	23.011684	41.007968	67.931664
gl	float64	1	0.162075	355	0.001129	21.978	0.124392	0.337827	21.978
class	int64	0	0.000000	2	0.0	1.0	0.0	0.0	0.0

- As for the features themselves, all of them are anonymized. So we unfortunately don't have any insight into what the health characteristics are.
- All of our features are floats except for EJ. It is a binary class with either an A or a B for its value which lead us to one hot encode it. We will touch more upon this feature later in the presentation when we discuss fairness.

EDA – Feature Distributions

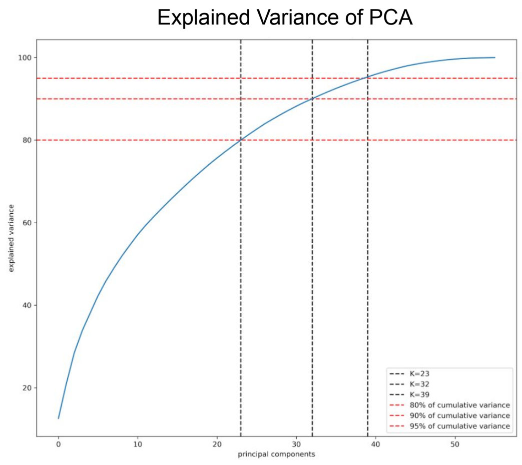
- Disproportion of label 0 and 1 observed across all features

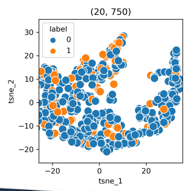


Here we can visually see the imbalance of the target class for all 56 features of the data set. We noticed a few bimodal features here such as CW, EL, and EJ. As mentioned previously, since EJ is binary we decided to focus on it for our fairness analysis.

Preprocessing

- Null values:
  - Occur in nine of the features
  - KNNImputer were used to fill the null values
- Separation of data:
  - Data was split into 80/20 for train and test
- Standardization:
  - sklearn.StandardScaler to scale using mean and standard deviation
- Feature reduction
  - Tested PCA and t-SNE
  - Maintains at least 80% of the variance





Berkeley
UNIVERSITY OF CALIFORNIA

One of our first steps in our preprocessing was our null analysis.

Our goal was to preserve as many observations as possible. In order to do this, we decided early on to augment the null values instead of dropping them. While 9 of the features had missing values, BQ and EL were the biggest offenders both of which are missing 60 values which is just under 10% of the total number of rows. Since the features are anonymized, we used KNNImputer with n_neighbors equal to two to fill in these missing values.

For the train/test datasets we tried both an 80/20 and a 70/30 split with train_test_split() from sklearn and found our models performed better with the 80/20 split

Due to the range of values in the all the features of the set, we also decided to use sklearn's StandardScaler

Finally, for feature reduction, we tested both PCA and tSNE in order to assist us in dimensionality reduction.

variance, but we also tested for 90 and 95% as seen by the redlines on the graph. In order to maintain 80% we would need at least 23 features; 32 for 90%; and 39 for 95%. We decided to go with 23 features as we felt it was a good balance between reduction and maintaining a strong variance with the original data

We also used tSNE although it is mainly used for dataviz. We felt that it might prove helpful to try reducing our dimensionality with it. However as seen in the graph in the lower right, we were unable to achieve linear separability using tSNE. This graph is one example from our tSNE testing of multiple different perplexities and iterations. Perplexity determines how many neighborhoods are near the center and iterations is simply just the number of times the model ran. This specific tSNE model is one with a perplexity of 20 that had 750 iterations.

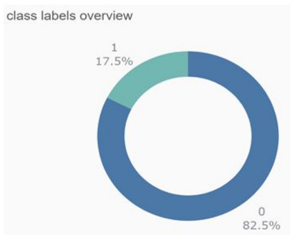
Baseline Model - Majority Vote

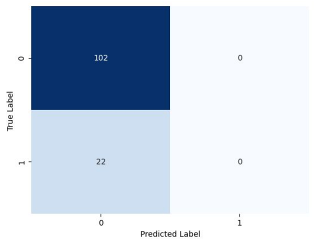
```
from sklearn.dummy import DummyClassifier

dummy_clf = DummyClassifier(strategy="most_frequent")
dummy_clf.fit(X_train, y_train)
print('Majority vote baseline train/test accuracies %.3f/%.3f' %
      (dummy_clf.score(X_train, y_train), dummy_clf.score(X_dev, y_dev)))
```

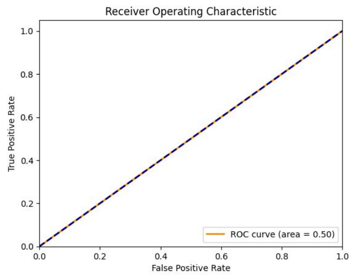
```
Majority vote baseline train/test accuracies 0.826/0.823
```

- Majority voting - selecting the most frequent label
- No learning from features, merely reflecting the distribution of our dataset
- Yields a 82% accuracy due to imbalanced datasets – benchmark for more sophisticated models

```
ROC AUC:  0.5
```



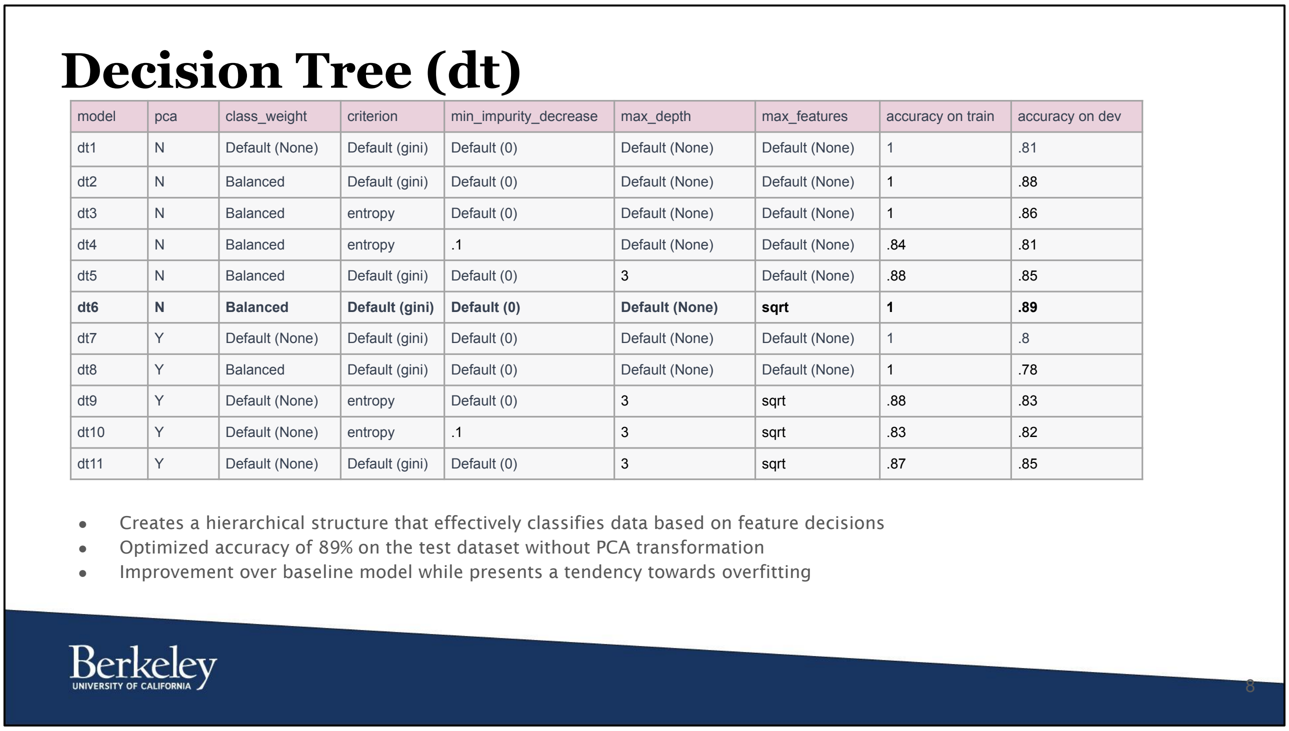




Berkeley
UNIVERSITY OF CALIFORNIA

●The baseline we used is a majority vote model that always predicts the most frequent class in the train data, which is the -ve class and it accounts for 83% of the data as shown in the pie chart here. Although this approach does not learn any features but due to the imbalanced dataset we have, the accuracy of 82% on test data is quite decent matching the percentage of the -ve class in the data. But when we look at confusion matrix, it's not performing so well as the ROC curve is essentially a diagonal line indicating the model does no good than random guessing, and as a result the AUC score is only .5.

●However this baseline model provides a benchmark against which we can compare more sophisticated models including decision tree, ensembles and neural network models. Next slide please.



Decision Tree (dt)

model	pca	class_weight	criterion	min_impurity_decrease	max_depth	max_features	accuracy on train	accuracy on dev
dt1	N	Default (None)	Default (gini)	Default (0)	Default (None)	Default (None)	1	.81
dt2	N	Balanced	Default (gini)	Default (0)	Default (None)	Default (None)	1	.88
dt3	N	Balanced	entropy	Default (0)	Default (None)	Default (None)	1	.86
dt4	N	Balanced	entropy	.1	Default (None)	Default (None)	.84	.81
dt5	N	Balanced	Default (gini)	Default (0)	3	Default (None)	.88	.85
dt6	**N**	**Balanced**	**Default (gini)**	**Default (0)**	**Default (None)**	**sqrt**	**1**	**.89**
dt7	Y	Default (None)	Default (gini)	Default (0)	Default (None)	Default (None)	1	.8
dt8	Y	Balanced	Default (gini)	Default (0)	Default (None)	Default (None)	1	.78
dt9	Y	Default (None)	entropy	Default (0)	3	sqrt	.88	.83
dt10	Y	Default (None)	entropy	.1	3	sqrt	.83	.82
dt11	Y	Default (None)	Default (gini)	Default (0)	3	sqrt	.87	.85

- Creates a hierarchical structure that effectively classifies data based on feature decisions
- Optimized accuracy of 89% on the test dataset without PCA transformation
- Improvement over baseline model while presents a tendency towards overfitting

- Here’s an ablation table showing all the hyperparameters we’ve tuned for decision tree. The best performer is model 6 with a balanced class weight and maximum #features (for best split) set to be square root of total #features and all else as default. Noticeably this model did not perform PCA – the reason we suspect could be that PCA transformation could lose interpretability of the original features and might not be effective in dealing with non-linear data which is our case here. Next slide please.

Decision Tree *(cont.)*

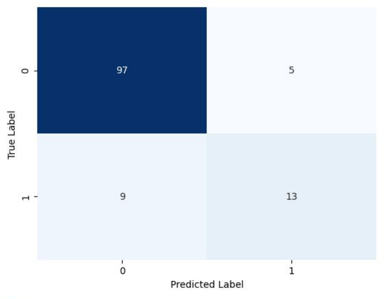
Training Data: 493, 24
Test Data: 124, 24

Best Performer

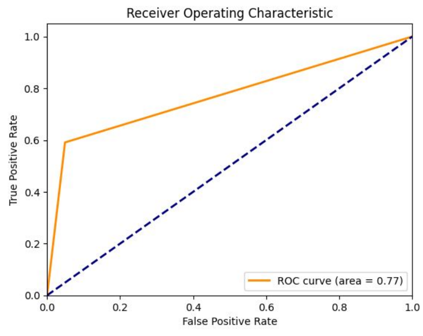
- data before pca transformation
- class_weight = 'balanced'
- criterion = gini
- min_impurity_decrease = 0
- max_depth = None
- max_features = sqrt

Model Evaluation:

```
Train/test accuracies 1.000/0.887
 Precision score: 0.819
 Recall score: 0.771
 F1 score 0.791
```



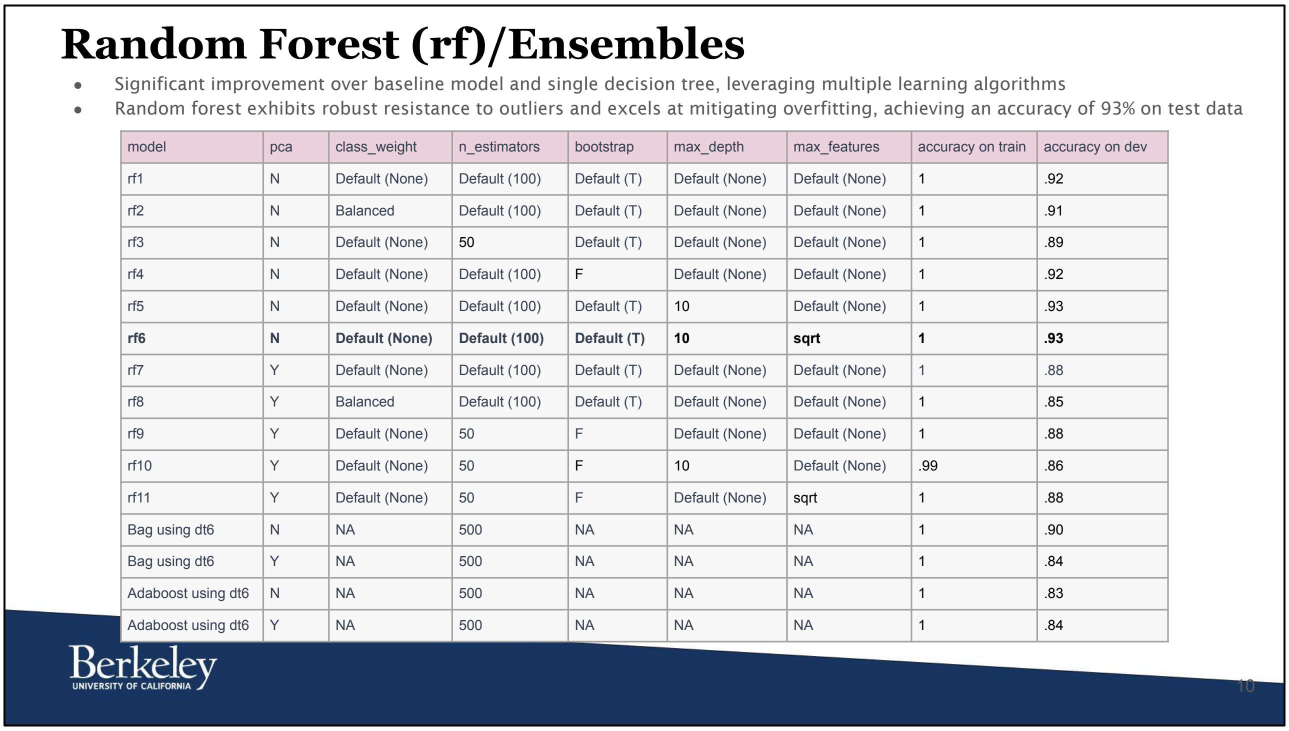
```
ROC AUC:  0.771
```



Berkeley UNIVERSITY OF CALIFORNIA

●With this, we achieved an accuracy of 1 on train data and 89% on test data. While this implies potential overfitting, both accuracy and AUC improved notably over the baseline model. ROC curve is above the diagonal line now, suggesting the model is better at distinguishing btw the 2 classes than just random guessing.

●To address overfitting and further improve on our model performances, we tried ensembles using random forest, bagging and adaboost classifers (all from sklearn). Next slide please.



Random Forest (rf)/Ensembles

- Significant improvement over baseline model and single decision tree, leveraging multiple learning algorithms
- Random forest exhibits robust resistance to outliers and excels at mitigating overfitting, achieving an accuracy of 93% on test data

model	pca	class_weight	n_estimators	bootstrap	max_depth	max_features	accuracy on train	accuracy on dev
rf1	N	Default (None)	Default (100)	Default (T)	Default (None)	Default (None)	1	.92
rf2	N	Balanced	Default (100)	Default (T)	Default (None)	Default (None)	1	.91
rf3	N	Default (None)	50	Default (T)	Default (None)	Default (None)	1	.89
rf4	N	Default (None)	Default (100)	F	Default (None)	Default (None)	1	.92
rf5	N	Default (None)	Default (100)	Default (T)	10	Default (None)	1	.93
rf6	**N**	**Default (None)**	**Default (100)**	**Default (T)**	**10**	**sqrt**	**1**	**.93**
rf7	Y	Default (None)	Default (100)	Default (T)	Default (None)	Default (None)	1	.88
rf8	Y	Balanced	Default (100)	Default (T)	Default (None)	Default (None)	1	.85
rf9	Y	Default (None)	50	F	Default (None)	Default (None)	1	.88
rf10	Y	Default (None)	50	F	10	Default (None)	.99	.86
rf11	Y	Default (None)	50	F	Default (None)	sqrt	1	.88
Bag using dt6	N	NA	500	NA	NA	NA	1	.90
Bag using dt6	Y	NA	500	NA	NA	NA	1	.84
Adaboost using dt6	N	NA	500	NA	NA	NA	1	.83
Adaboost using dt6	Y	NA	500	NA	NA	NA	1	.84

Berkeley
UNIVERSITY OF CALIFORNIA

- As shown in another ablation table here, the best performer is random forest with a max depth of 10 and max #features (for best split) as sqrt of the total number. This did not use PCA either and we achieved an accuracy of 1 on train data and 93% on test. Next slide please.

Random Forest/Ensembles *(cont.)*

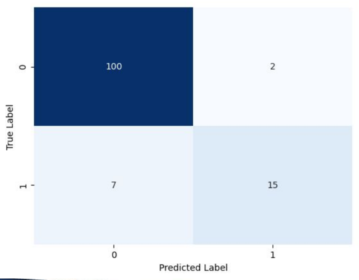
Training Data: 493, 24
Test Data: 124, 24

Best Performer

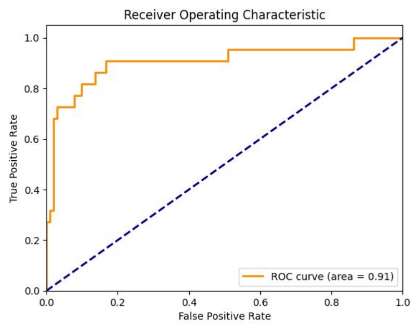
- data before pca transformation
- class_weight = 'balanced'
- n_estimators = 100
- bootstrap = True
- max_depth = 10
- max_features = sqrt

Model Evaluation:

```
Train/test accuracies 1.000/0.927
Precision score: 0.908
Recall score: 0.831
F1 score 0.863
```

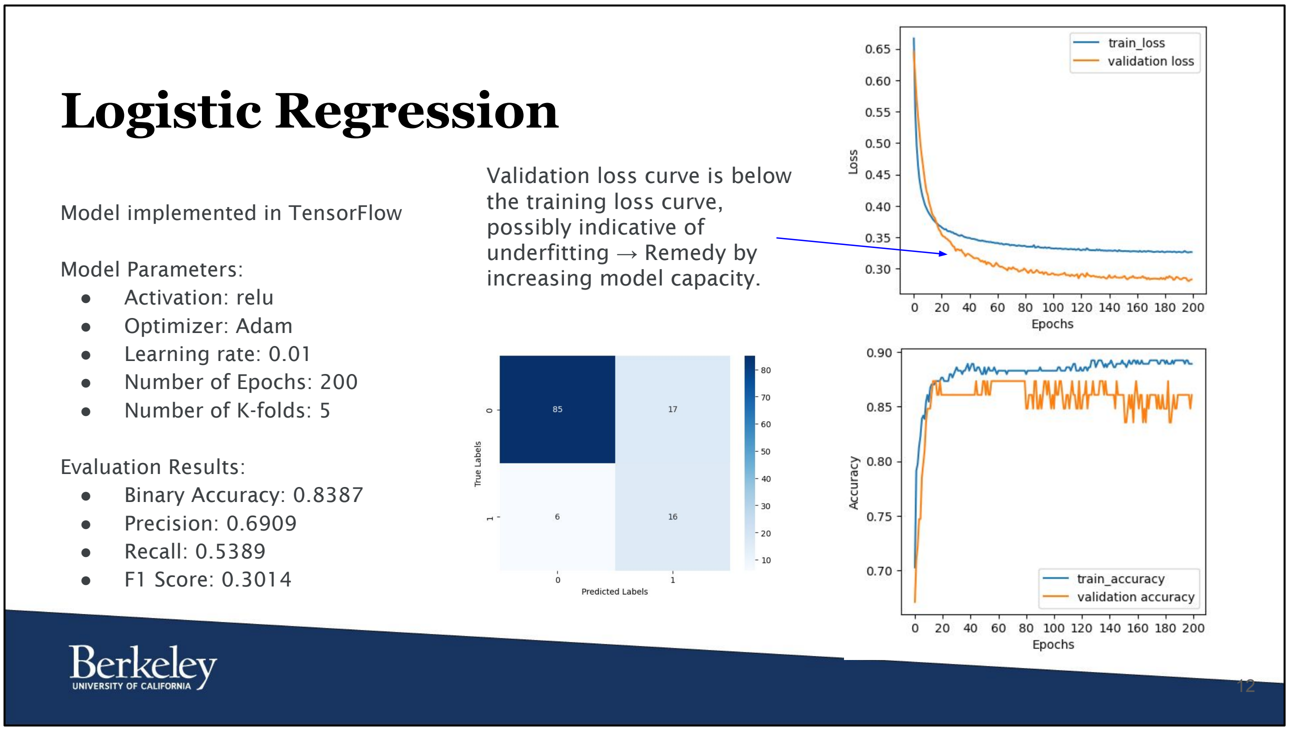


```
ROC AUC:  0.907
```



Berkeley
UNIVERSITY OF CALIFORNIA

- With reduced likelihood of overfitting, both accuracy and performance metrics have shown significant improvement over the baseline and the decision trees. AUC of .91 suggests a strong ability of our model to distinguish btw 2 classes, which is the best we've achieved so far.



Logistic Regression

Model implemented in TensorFlow

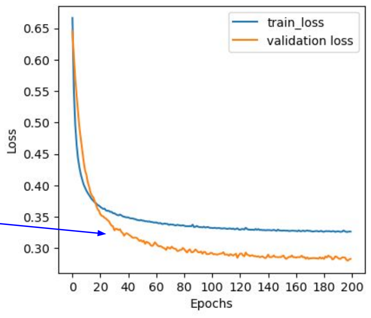
Model Parameters:

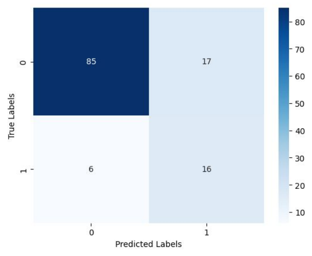
- Activation: relu
- Optimizer: Adam
- Learning rate: 0.01
- Number of Epochs: 200
- Number of K-folds: 5

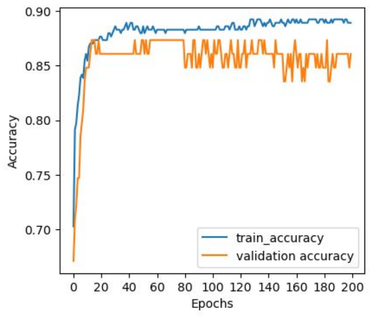
Evaluation Results:

- Binary Accuracy: 0.8387
- Precision: 0.6909
- Recall: 0.5389
- F1 Score: 0.3014

Validation loss curve is below the training loss curve, possibly indicative of underfitting → Remedy by increasing model capacity.







Berkeley UNIVERSITY OF CALIFORNIA

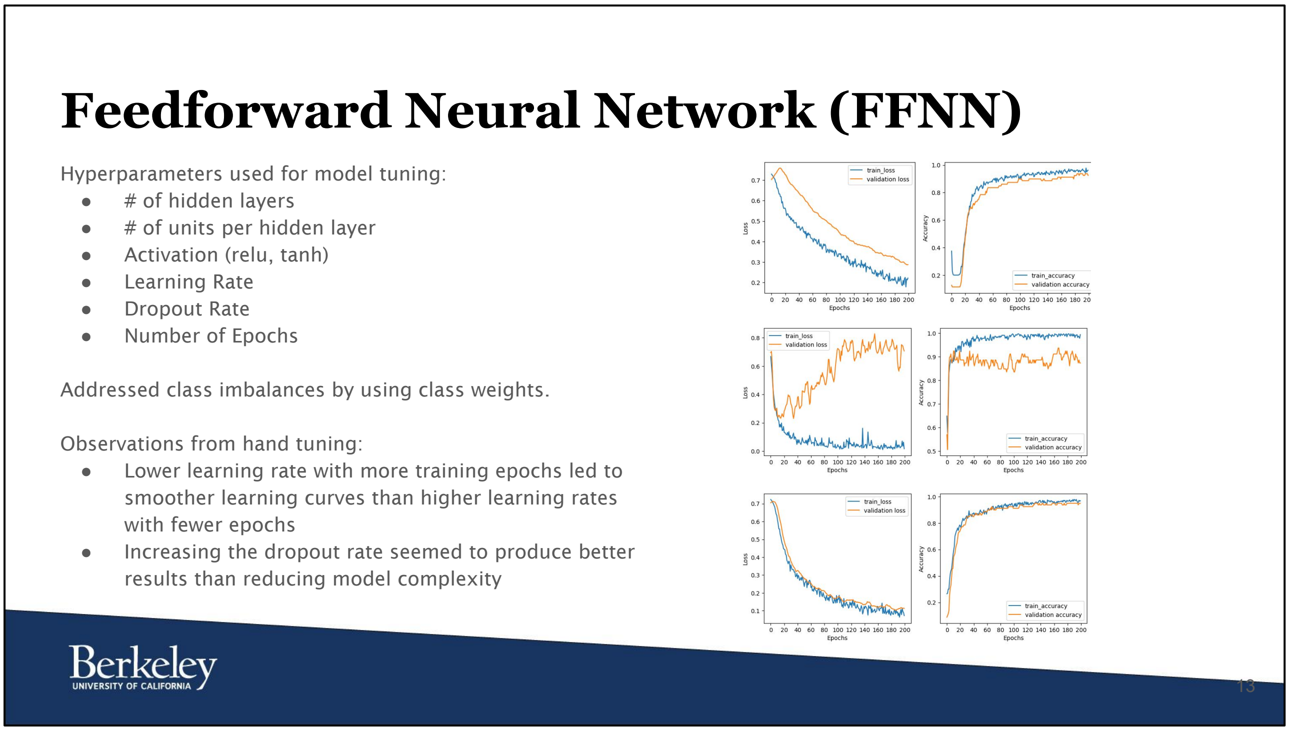
· I will be covering Logistic Regression and then quickly move on to Feedforward Neural Networks with Akhila.

· For logistic regression, we implemented the model using TensorFlow's Sequential API.

· For this model, we used relu activation with the Adam optimizer. We applied a learning rate of 0.01 and ran the model for 200 Epochs with k-folds cross validation set to 5 folds.

· This model produced a binary accuracy of 0.839, which is only a very slight improvement over the baseline model's performance of 0.826.

· When examining our training curves, we noted that the validation loss curve was below the training loss curve, which indicated to us that the model could possibly be underfitting the dataset. As we discussed in live session, one possible remedy for this type of underfitting is to increase the model capacity, which we pursued by moving on to a Feedforward Neural Network model.



Feedforward Neural Network (FFNN)

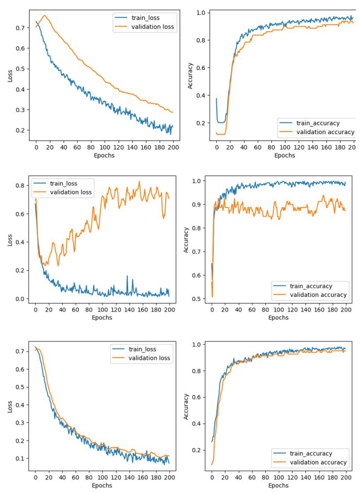
Hyperparameters used for model tuning:

- # of hidden layers
- # of units per hidden layer
- Activation (relu, tanh)
- Learning Rate
- Dropout Rate
- Number of Epochs

Addressed class imbalances by using class weights.

Observations from hand tuning:

- Lower learning rate with more training epochs led to smoother learning curves than higher learning rates with fewer epochs
- Increasing the dropout rate seemed to produce better results than reducing model complexity



Berkeley UNIVERSITY OF CALIFORNIA

· For our Feedforward Neural Network, we added dense layers and dropout layers to our model in TensorFlow.

· We tuned the model by modifying the number of hidden layers, the number of units per hidden layer, the activation method, the learning rate, the dropout rate, and the number of epochs.

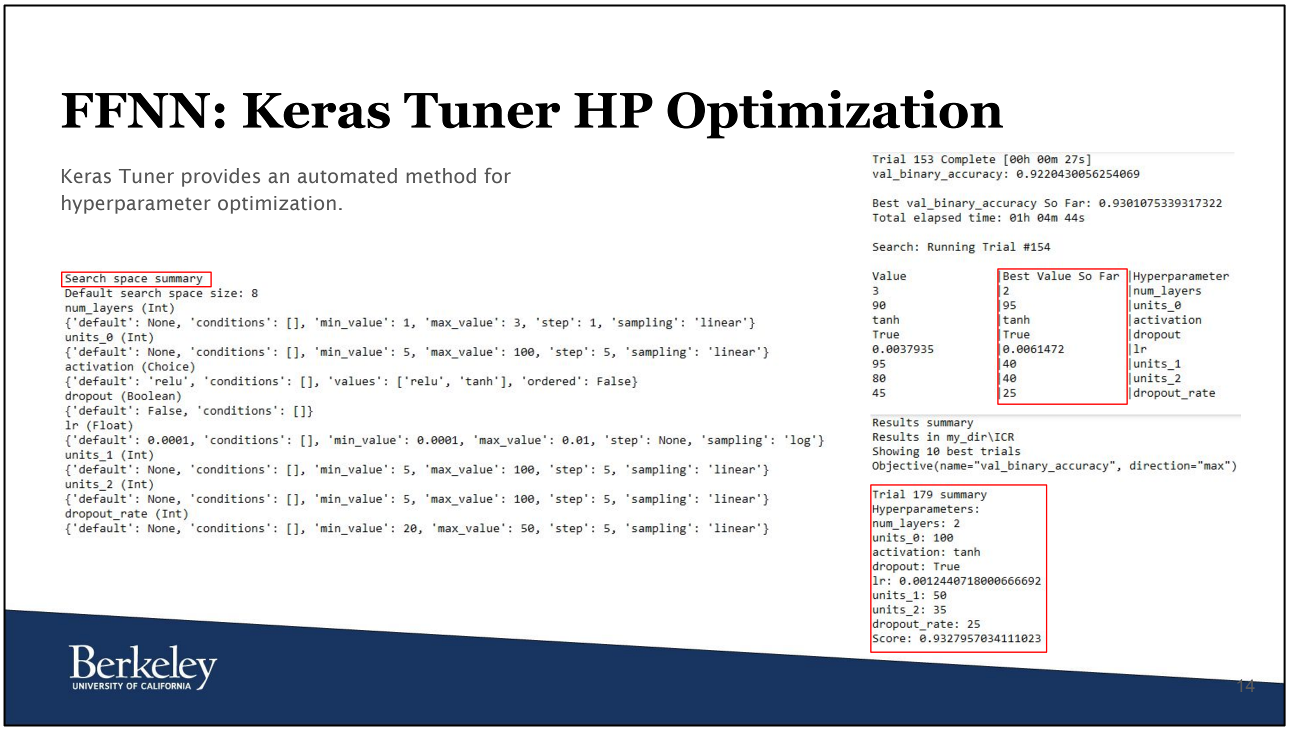
· To address the class imbalance that we identified during EDA, we assigned weights to each of the classes.

· On the right-hand side, we have provided a sample of some of the learning curves that the model produced as we hand tuned the model and made adjustments to improve performance.

· A couple observations that we made during the hand tuning process for our dataset include:

o that using a lower learning rate with more training epochs generally produced smoother learning curves than when we boosted the learning rate with a smaller number of training epochs and

o that increasing the dropout rate seemed to produce better results than reducing the complexity of the model.



FFNN: Keras Tuner HP Optimization

Keras Tuner provides an automated method for hyperparameter optimization.

```
Search space summary
Default search space size: 8
num_layers (Int)
{'default': None, 'conditions': [], 'min_value': 1, 'max_value': 3, 'step': 1, 'sampling': 'linear'}
units_0 (Int)
{'default': None, 'conditions': [], 'min_value': 5, 'max_value': 100, 'step': 5, 'sampling': 'linear'}
activation (Choice)
{'default': 'relu', 'conditions': [], 'values': ['relu', 'tanh'], 'ordered': False}
dropout (Boolean)
{'default': False, 'conditions': []}
lr (Float)
{'default': 0.0001, 'conditions': [], 'min_value': 0.0001, 'max_value': 0.01, 'step': None, 'sampling': 'log'}
units_1 (Int)
{'default': None, 'conditions': [], 'min_value': 5, 'max_value': 100, 'step': 5, 'sampling': 'linear'}
units_2 (Int)
{'default': None, 'conditions': [], 'min_value': 5, 'max_value': 100, 'step': 5, 'sampling': 'linear'}
dropout_rate (Int)
{'default': None, 'conditions': [], 'min_value': 20, 'max_value': 50, 'step': 5, 'sampling': 'linear'}
```

```
Trial 153 Complete [00h 00m 27s]
val_binary_accuracy: 0.9220430056254069

Best val_binary_accuracy So Far: 0.9301075339317322
Total elapsed time: 01h 04m 44s

Search: Running Trial #154

Value             |Best Value So Far |Hyperparameter
3                 |2                 |num_layers
90                |95                |units_0
tanh              |tanh              |activation
True              |True              |dropout
0.0037935         |0.0061472         |lr
95                |40                |units_1
80                |40                |units_2
45                |25                |dropout_rate
```

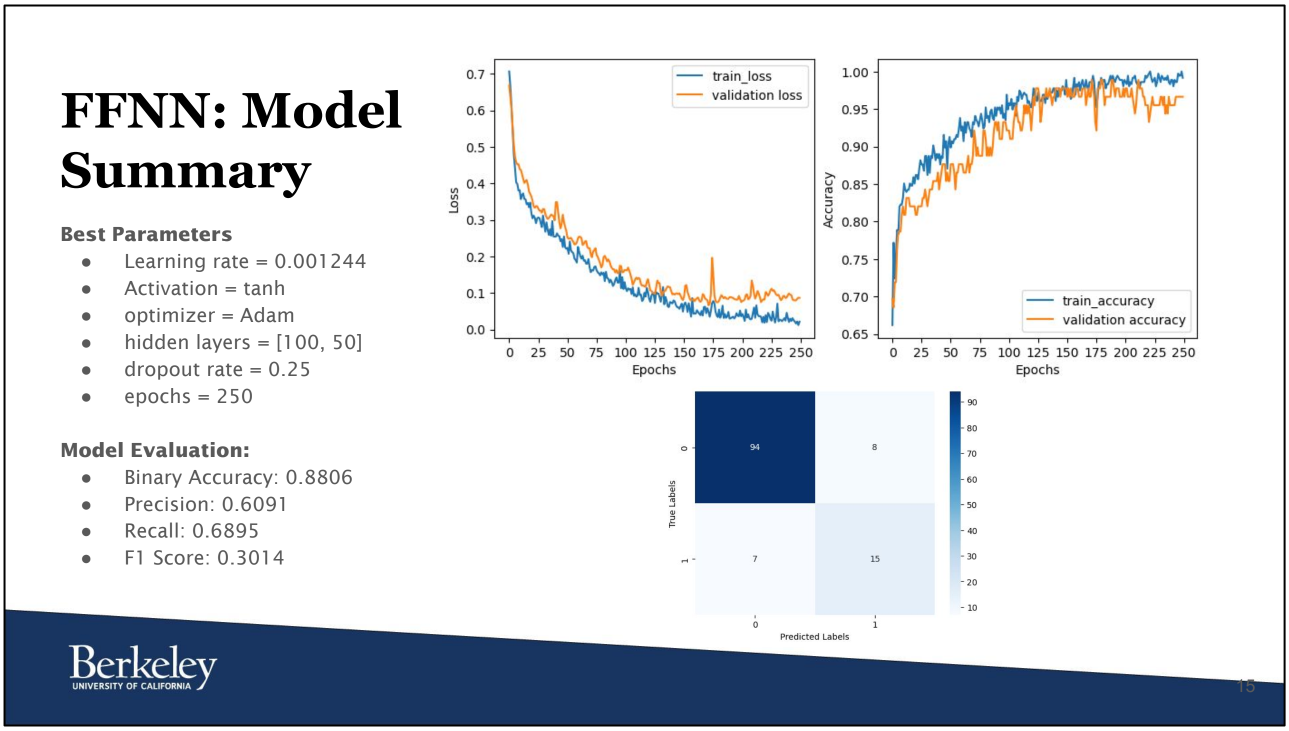
```
Results summary
Results in my_dir\ICR
Showing 10 best trials
Objective(name="val_binary_accuracy", direction="max")
```

```
Trial 179 summary
Hyperparameters:
num_layers: 2
units_0: 100
activation: tanh
dropout: True
lr: 0.0012440718000666692
units_1: 50
units_2: 35
dropout_rate: 25
Score: 0.9327957034111023
```

Berkeley
UNIVERSITY OF CALIFORNIA

· In addition to hand tuning the model, we attempted to using keras tuner, a hyperparameter optimizer, to see if a systematic approach would produce a materially better result.

- The optimizer operates by specifying the search space--left side of the slide--with ranges and step sizes as well as the number of trials and then allowing the optimizer to run.
- On the right side of the slide, example optimizer output is provided for reference.



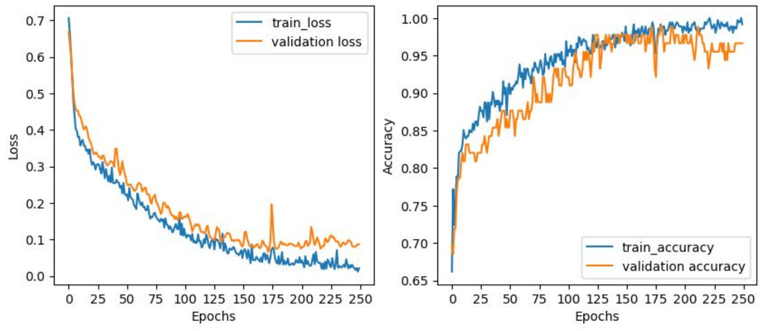
FFNN: Model Summary

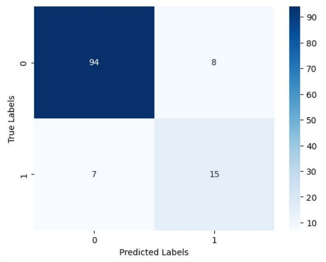
Best Parameters

- Learning rate = 0.001244
- Activation = tanh
- optimizer = Adam
- hidden layers = [100, 50]
- dropout rate = 0.25
- epochs = 250

Model Evaluation:

- Binary Accuracy: 0.8806
- Precision: 0.6091
- Recall: 0.6895
- F1 Score: 0.3014

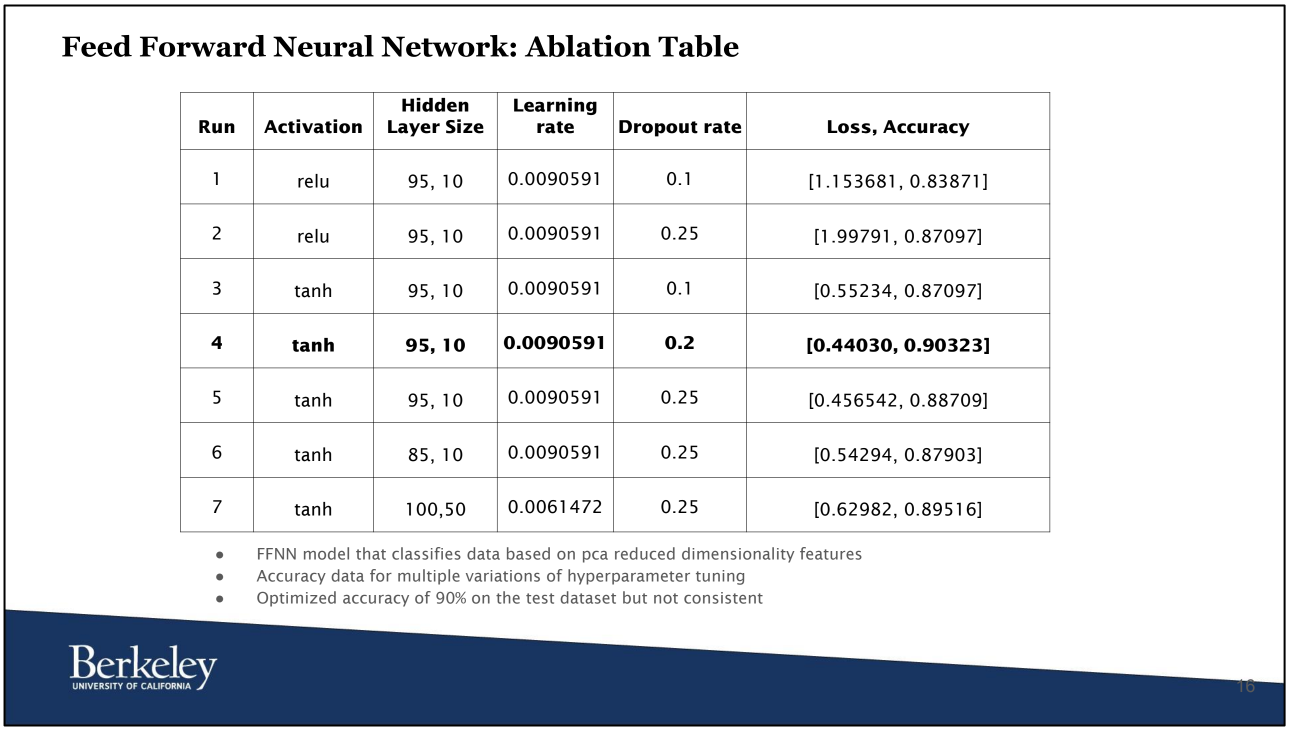




Berkeley UNIVERSITY OF CALIFORNIA

· Our best FFNN model produced a binary accuracy of about 0.881 with the parameters shown on the slide. This is a meaningful improvement over the 0.826 for the baseline model. Example learning curves for this test run as well as the confusion matrix are provided for reference. As the curves at the top of the slide indicate, the training curves are modestly better than the validation curves, which demonstrates that we achieved the slight overfitting that we were aiming for in the tuning process.

· Next, Akhila will discuss the ablation study for this model.



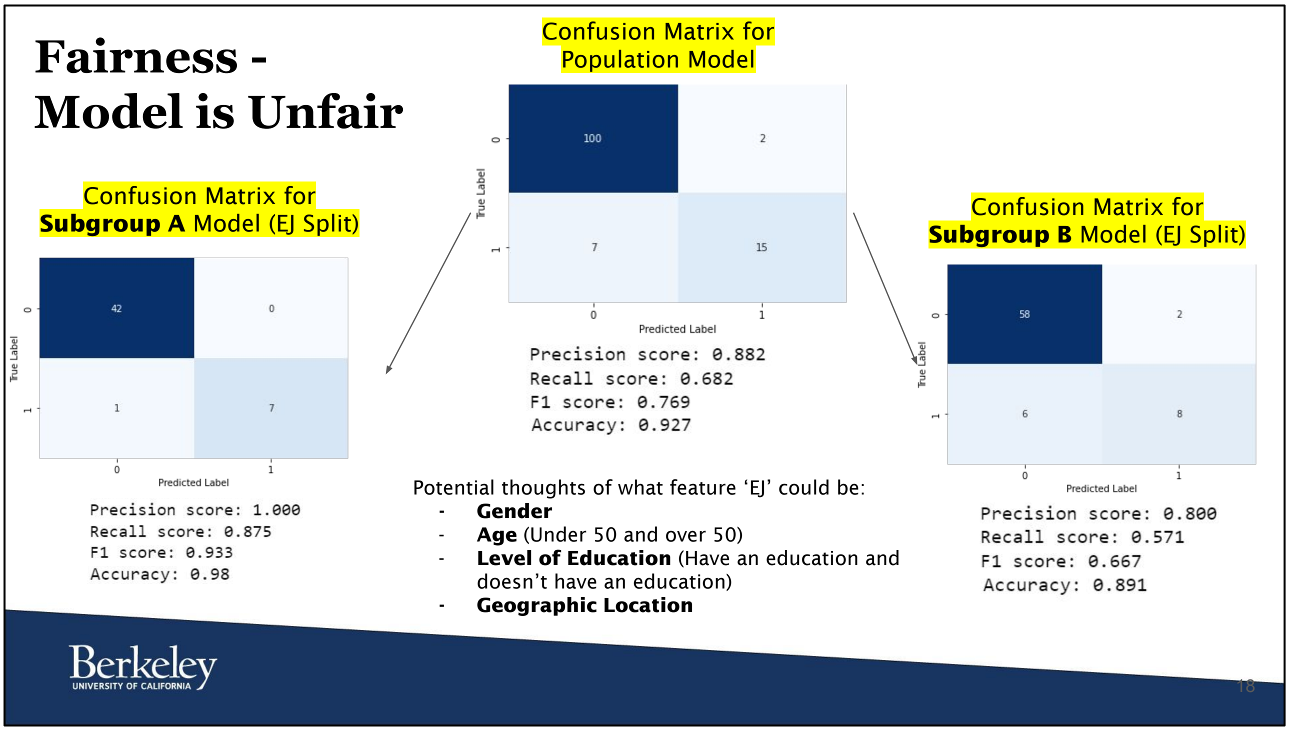
Feed Forward Neural Network: Ablation Table

Run	Activation	Hidden Layer Size	Learning rate	Dropout rate	Loss, Accuracy
1	relu	95, 10	0.0090591	0.1	[1.153681, 0.83871]
2	relu	95, 10	0.0090591	0.25	[1.99791, 0.87097]
3	tanh	95, 10	0.0090591	0.1	[0.55234, 0.87097]
4	**tanh**	**95, 10**	**0.0090591**	**0.2**	**[0.44030, 0.90323]**
5	tanh	95, 10	0.0090591	0.25	[0.456542, 0.88709]
6	tanh	85, 10	0.0090591	0.25	[0.54294, 0.87903]
7	tanh	100,50	0.0061472	0.25	[0.62982, 0.89516]

- FFNN model that classifies data based on pca reduced dimensionality features
- Accuracy data for multiple variations of hyperparameter tuning
- Optimized accuracy of 90% on the test dataset but not consistent

Berkeley
UNIVERSITY OF CALIFORNIA

Akhila



Fairness - Model is Unfair

Confusion Matrix for Population Model

True Label \ Predicted Label	0	1
0	100	2
1	7	15

```
Precision score: 0.882
Recall score: 0.682
F1 score: 0.769
Accuracy: 0.927
```

Confusion Matrix for **Subgroup A** Model (EJ Split)

True Label \ Predicted Label	0	1
0	42	0
1	1	7

```
Precision score: 1.000
Recall score: 0.875
F1 score: 0.933
Accuracy: 0.98
```

Confusion Matrix for **Subgroup B** Model (EJ Split)

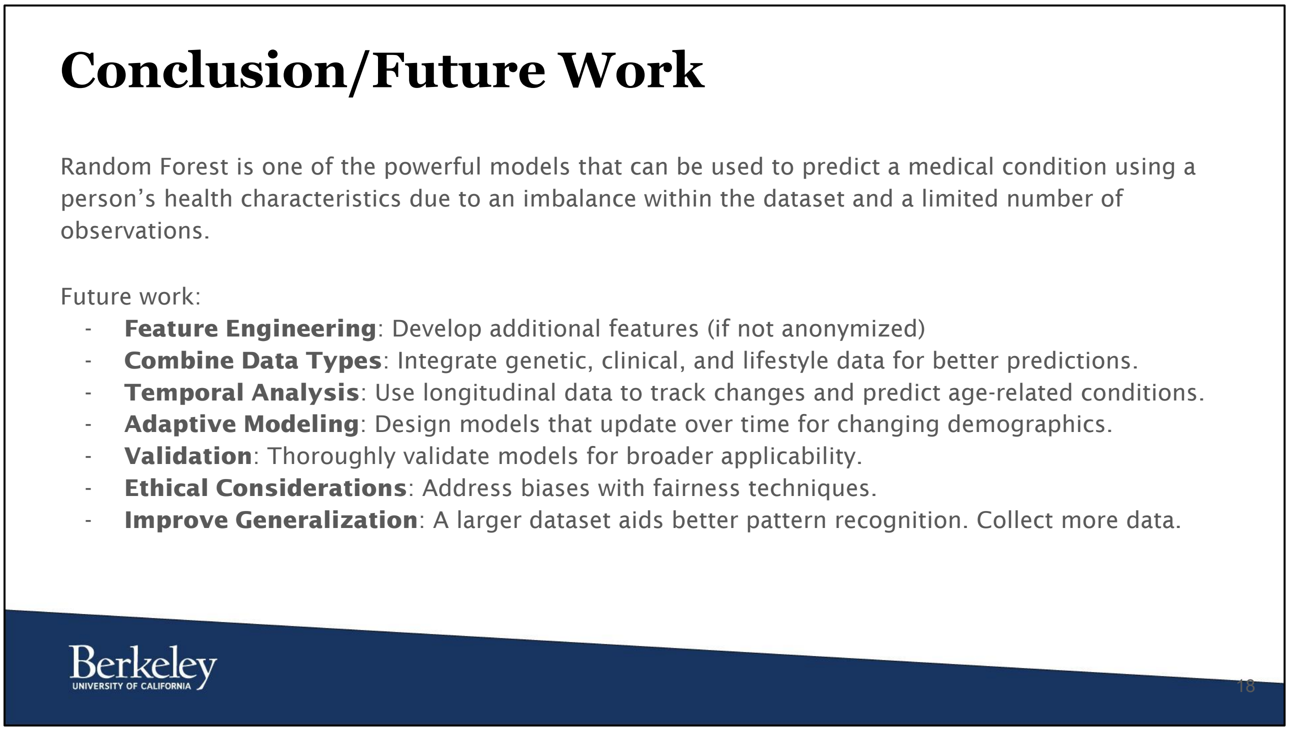
True Label \ Predicted Label	0	1
0	58	2
1	6	8

```
Precision score: 0.800
Recall score: 0.571
F1 score: 0.667
Accuracy: 0.891
```

Potential thoughts of what feature 'EJ' could be:
- **Gender**
- **Age** (Under 50 and over 50)
- **Level of Education** (Have an education and doesn't have an education)
- **Geographic Location**

Berkeley UNIVERSITY OF CALIFORNIA

As Kevin had mentioned, during our exploratory data analysis (EDA), we addressed potential biases in our dataset. One feature, labeled 'EJ,' caught our attention due to potential bias. As the features were anonymized, we could only speculate that 'EJ' might represent gender, education level, or age category, given its non-numeric nature in contrast to other features. To assess fairness, we employed a Random Forest model with the highest accuracy (93%) from our evaluation. We partitioned the dataset into subgroups based on the protected feature ('EJ'). This involved creating separate subgroups for different values of 'EJ,' such as males and females if it represented gender. The performance of these subgroups was evaluated using precision, recall, and F1-score in comparison to the overall population model. Notably, Subgroup A's model exhibited higher precision and recall, indicating performance comparable to or better than the population model. However, Subgroup B's model showed slightly lower precision and recall, implying potential accuracy issues compared to the population model. While this suggests fairness concerns for Subgroup B, a more thorough analysis is necessary to confirm and quantify any fairness discrepancies. One way we could remedy this bias is apply post-processing techniques to adjust the predictions made by the model like recalibration which equalizes the predictions’ distribution across the different subgroups while maintaining the overall model accuracy.



Conclusion/Future Work

Random Forest is one of the powerful models that can be used to predict a medical condition using a person's health characteristics due to an imbalance within the dataset and a limited number of observations.

Future work:
- **Feature Engineering**: Develop additional features (if not anonymized)
- **Combine Data Types**: Integrate genetic, clinical, and lifestyle data for better predictions.
- **Temporal Analysis**: Use longitudinal data to track changes and predict age-related conditions.
- **Adaptive Modeling**: Design models that update over time for changing demographics.
- **Validation**: Thoroughly validate models for broader applicability.
- **Ethical Considerations**: Address biases with fairness techniques.
- **Improve Generalization**: A larger dataset aids better pattern recognition. Collect more data.

Berkeley
UNIVERSITY OF CALIFORNIA

To summarize our key results, Random Forest is one of the powerful models that can be used to predict a medical condition using a person's health characteristics due to an imbalance within the dataset and a limited number of observations. We learned that Train-Test splits significantly impacts model performance for high dimensional, small sample datasets.

If we were to continue our work and had more time, we could develop methods for combining multiple types of measurements, such as genetic, clinical, and lifestyle data, to enhance prediction accuracy. We would investigate the use of longitudinal data to monitor changes in anonymous characteristics over time and predict the progression of age-related conditions. We could continue to design models that can adapt and update over time as new data becomes available, ensuring that the model remains accurate as demographics and conditions change. We would rigorously validate the models on diverse and representative datasets to ensure their generalizability and robustness. We would also be able to address potential biases in the data that could lead to unfair or discriminatory predictions, especially in age-related conditions where healthcare disparities might be present. This includes implementing fairness-aware machine learning techniques to ensure equitable treatment across different demographic groups. And last, maybe a larger dataset can help our machine learning model learn more representative patterns and relationships present in the data.

Questions

Contributions

Akhila Ganti: Introduction, EDA, Fairness, FFNN Ablation Table
Megan Nguyen: EDA, Fairness Analysis, Random Forests, Conclusion
Leslie Nie: EDA, Decision Tree, Random Forests
Kevin Stallone: EDA, Preprocessing, Coding and Research
Tim Tung: EDA, Preprocessing, Logistic Regression/FFNN