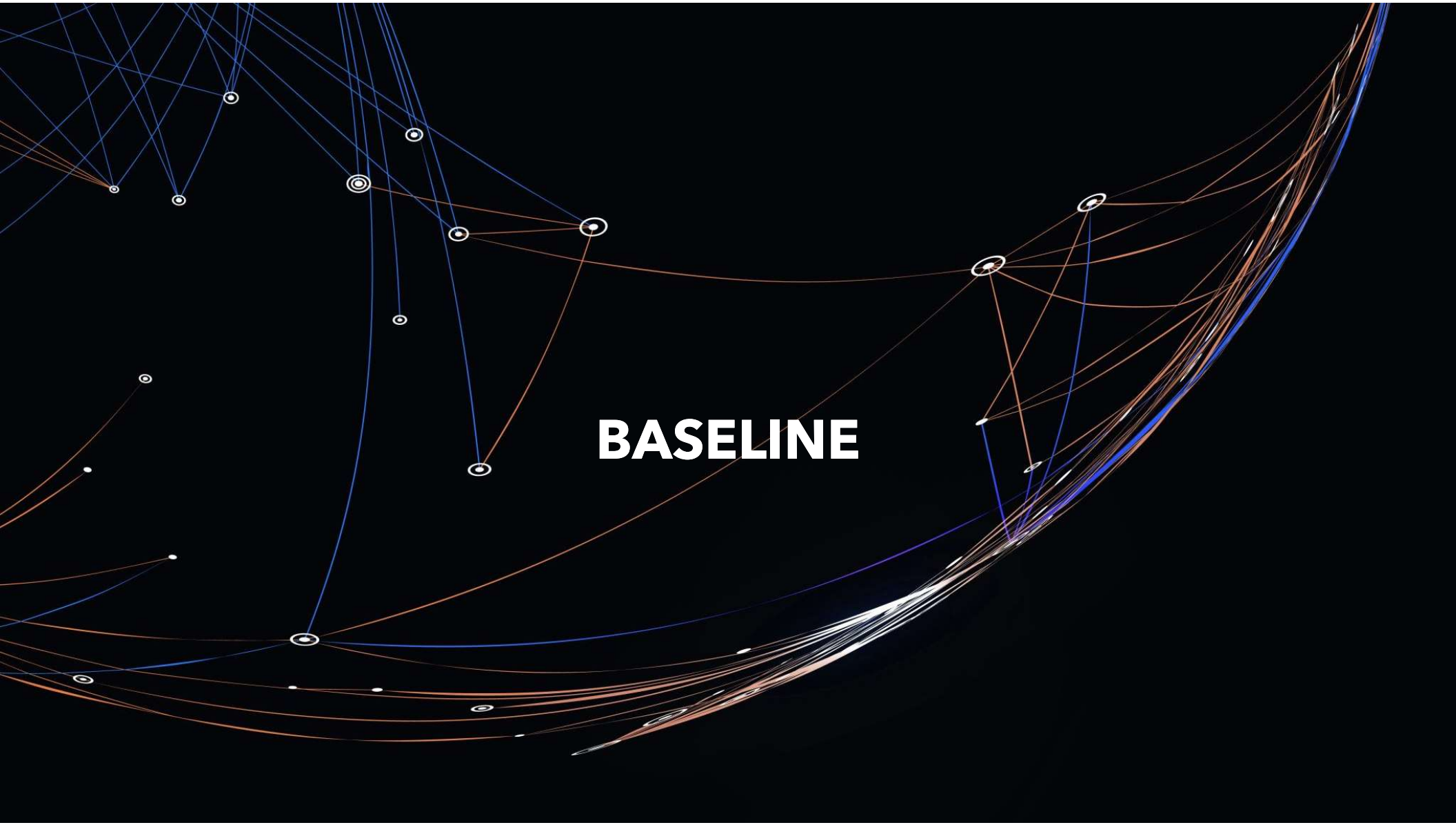
RANDOM FOREST

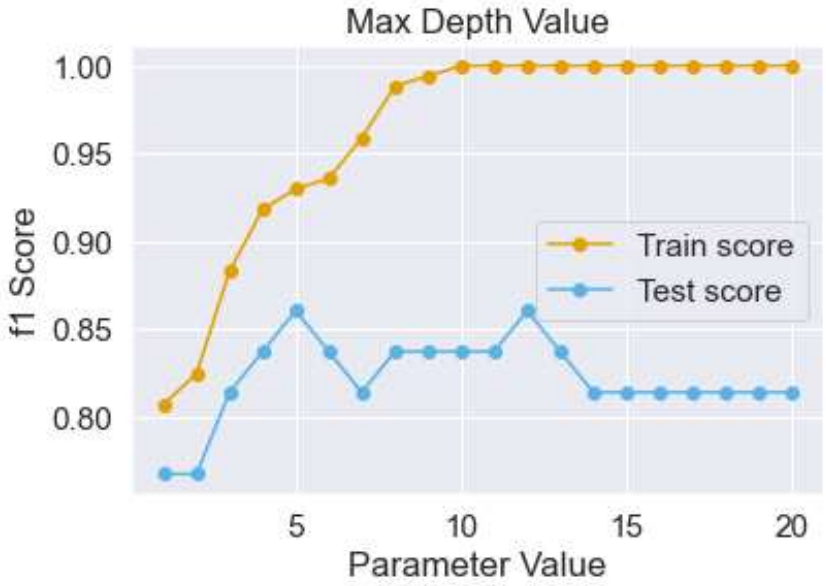
GROUP 5: KRIPANJALI DHUNGANA, JACK LYNN, NORMAN MORRIS, & SAM WAINRIGHT

INTRODUCTION

- Glass Identification dataset: **Can you correctly identify glass type?**
- Attribute Information: **RI, Na, Mg, Al, Si, K, Ca, Ba, Fe**
- Baseline models: **k-NN** and **Random Forest**
- Optimization models:
  - **Manual Hyperparameter** Training
  - **GridSearch Hyperparameter** Training
  - **Limited Input Columns** Training


BASELINE

	precision	recall	f1-score	support
Building	0.81	0.93	0.87	28
Container	1.00	0.50	0.67	2
Headlamp	0.86	1.00	0.92	6
Tableware	1.00	1.00	1.00	2
Vehicle	0.00	0.00	0.00	5
accuracy			0.81	43
macro avg	0.73	0.69	0.69	43
weighted avg	0.74	0.81	0.77	43

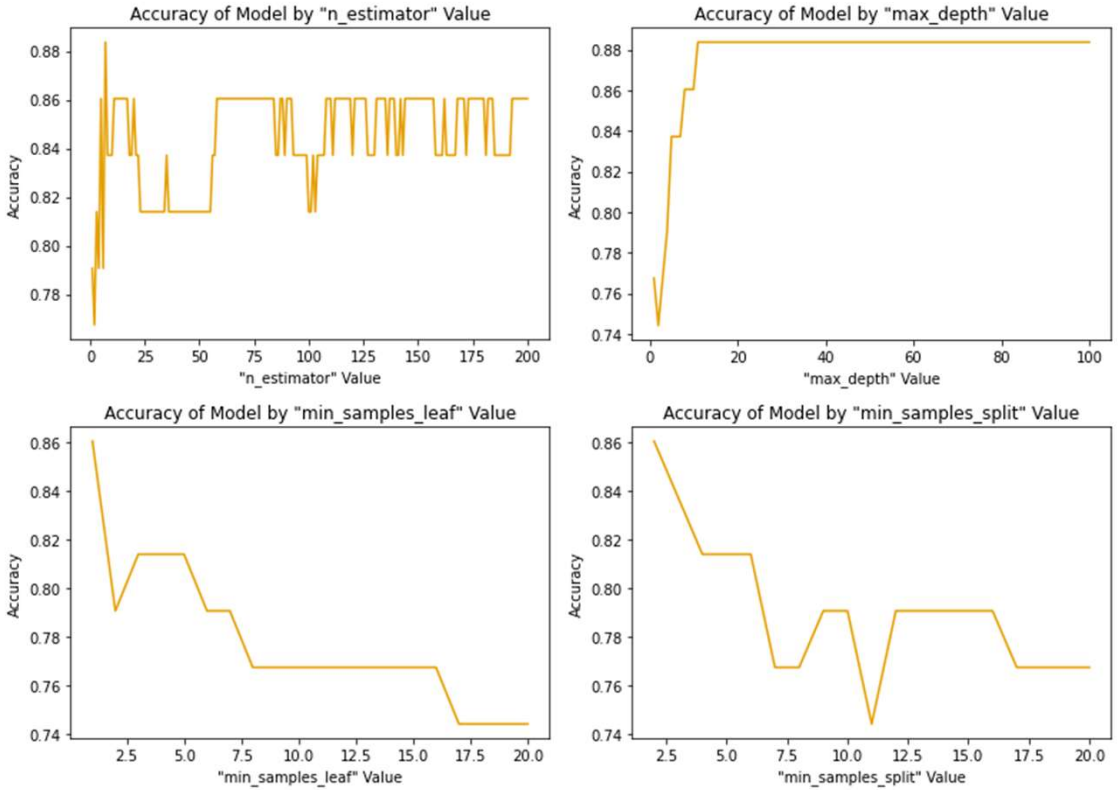


BASELINE:
UNALTERED
RANDOM FOREST

- Default hyperparameters set n_estimators = 100; random_state = 0; max_depth = 2
- Compare max depth 2 on x axis


OPTIMIZATION

MODEL 1: MANUAL HYPERPARAMETER TRAINING



- Held **other hyperparameters constant** while **varying one** (n_estimator, max_depth, min_samples_leaf, & min_samples_split)
- Evaluated for **highest accuracy**
- **More time efficient** (~1 min) than **GridSearch** (2 min to days)

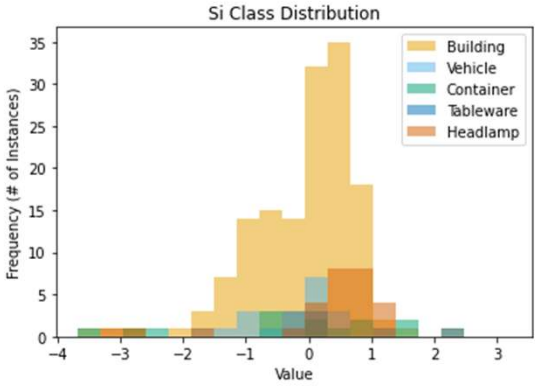
MODEL 2: GRIDSEARCH HYPERPARAMETER TRAINING

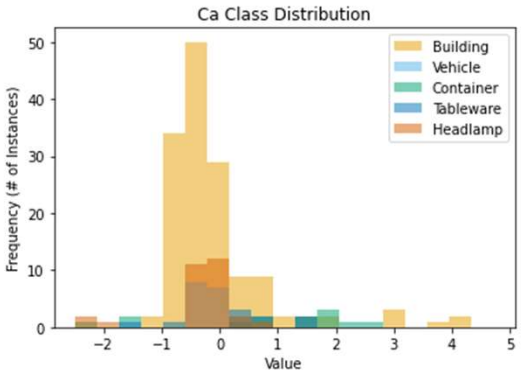
```
# Add tuned random forest model using grid search

# List Hyperparameters that we want to tune.
n_estimators = range(10, 251, 10)
max_features = ['sqrt'] # Not explored
max_depth = list(range(1, 21))
min_samples_leaf = [2] # Not explored
min_samples_split = [2] # Not explored

# Convert to dictionary
hyperparameters = dict()
hyperparameters['n_estimators'] = n_estimators
hyperparameters['max_features'] = max_features
hyperparameters['max_depth'] = max_depth
hyperparameters['min_samples_leaf'] = min_samples_leaf
hyperparameters['min_samples_split'] = min_samples_split
hyperparameters['criterion'] = ["gini"]

# Create new KNN object
rf_opt = RandomForestClassifier(random_state=0)

# Use grid search to find the ideal hyperparamters
grid_search_rf = GridSearchCV(rf_opt, hyperparameters, cv=2, verbose=5)

# Fit the model
grid_search_rf.fit(X_train, y_train)

# Add model to dictionary
models['Random Forest (Grid Search Tuned)'] = grid_search_rf
```

- Execute **every variation of model** (n_estimator & max_depth)
- Evaluated for **highest GINI index**
- Gets **highest possible accuracy** at the **cost of time**

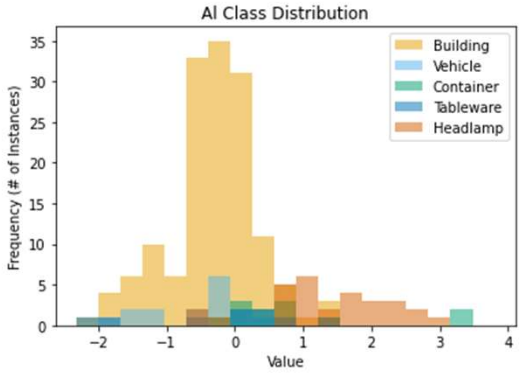
MODEL 3: LIMITED INPUT COLUMNS TRAINING

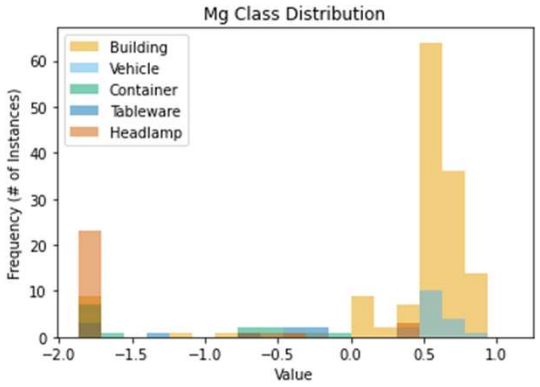
Example Dropped



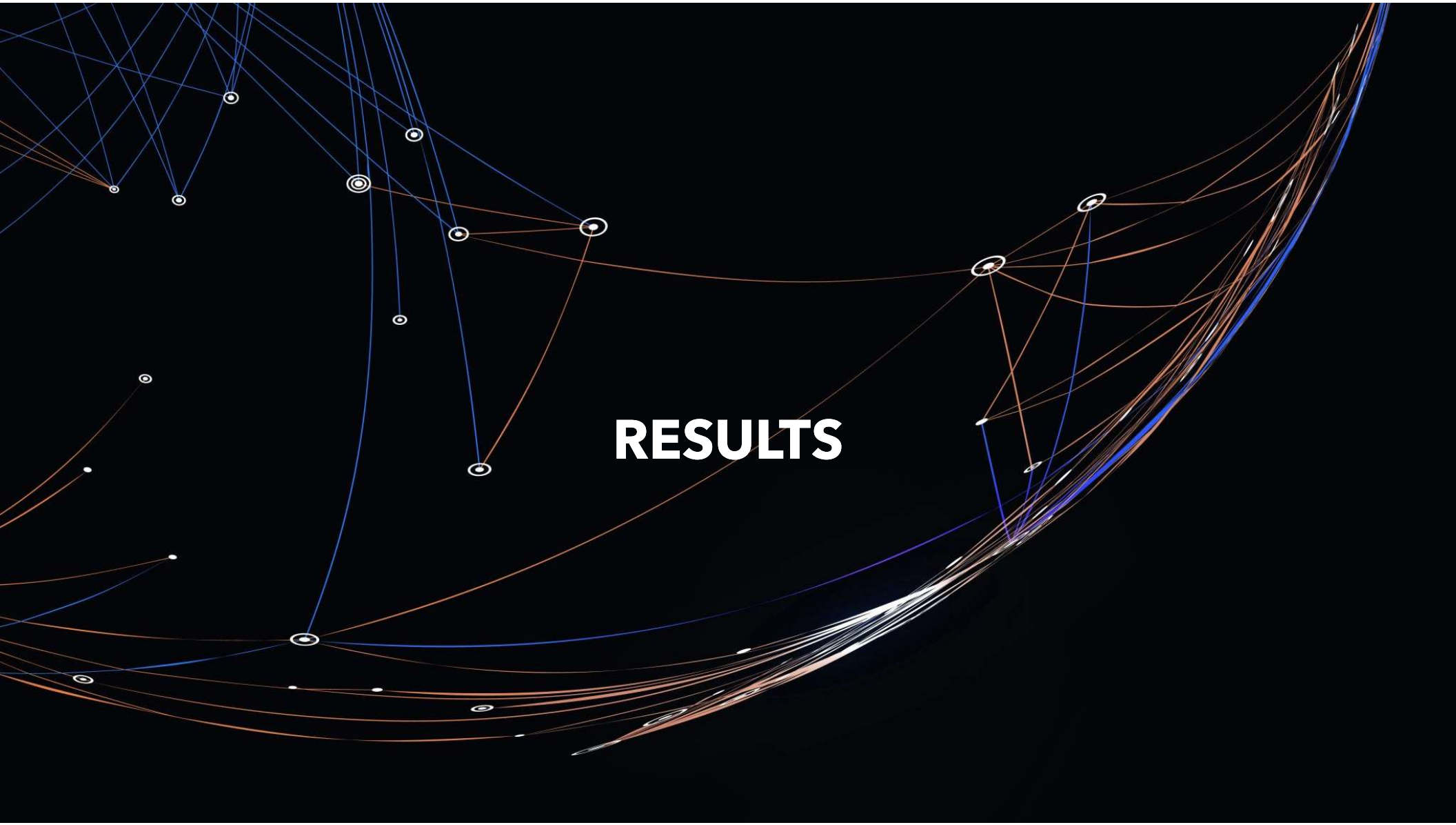


Example Retained



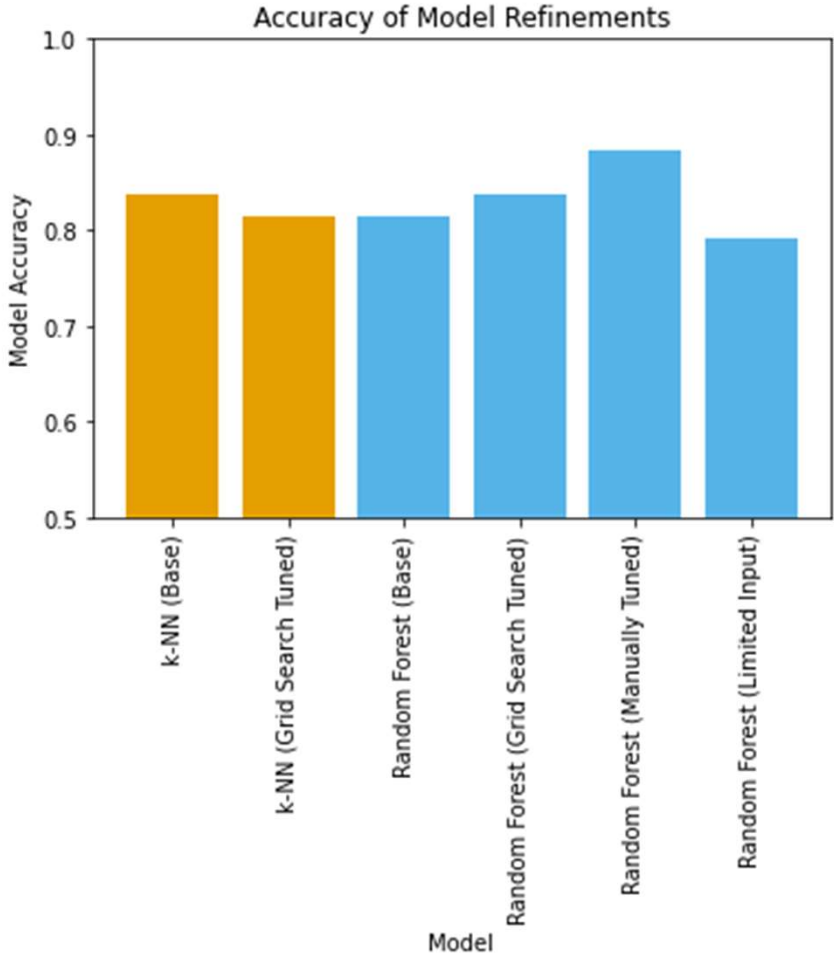


- **Dropped all variables** that do not have **sufficient distinction between classifications**
- Trained using **Model 1 hyperparameters**
- Potentially **remove variables** that contribute **unhelpful noise**

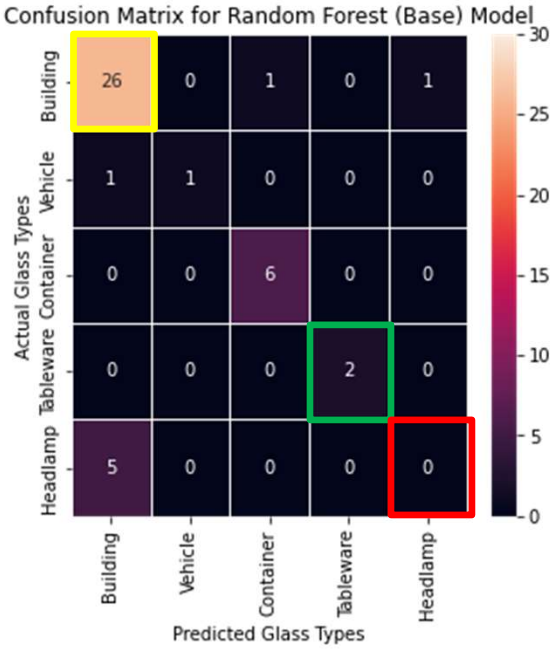

RESULTS

GridSearch and **manually tuned hyperparameters** of **Random Forest** models produced the **highest accuracy**

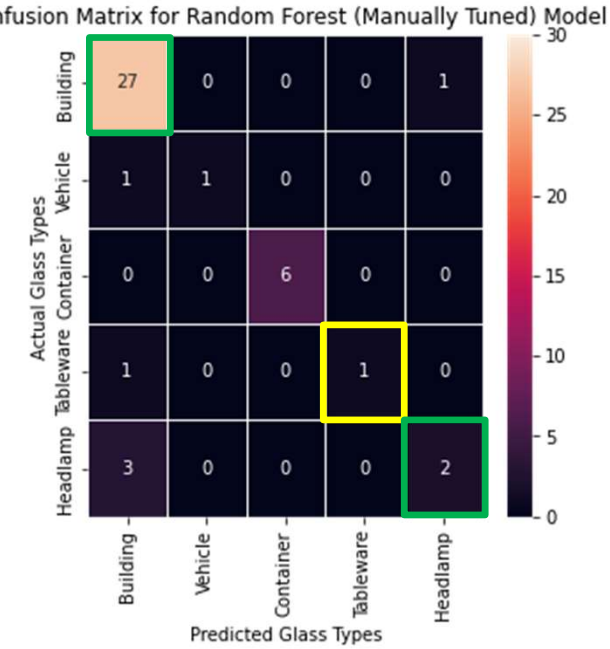
Removing data columns **reduced accuracy**



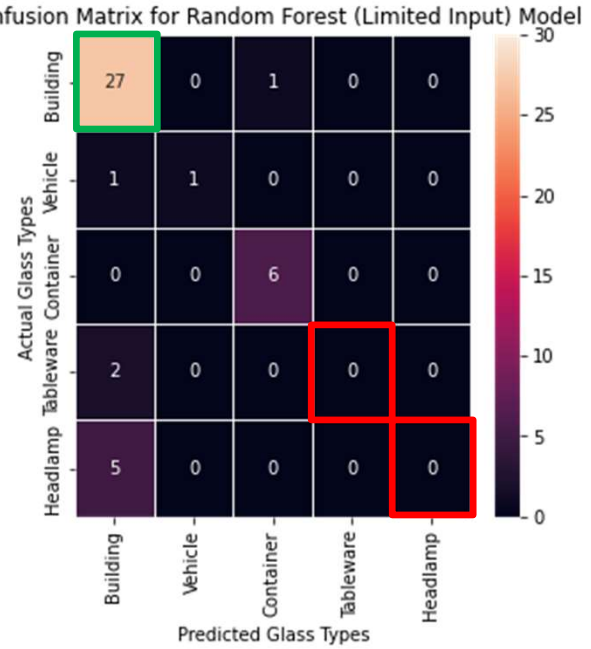
Base



Manually Tuned

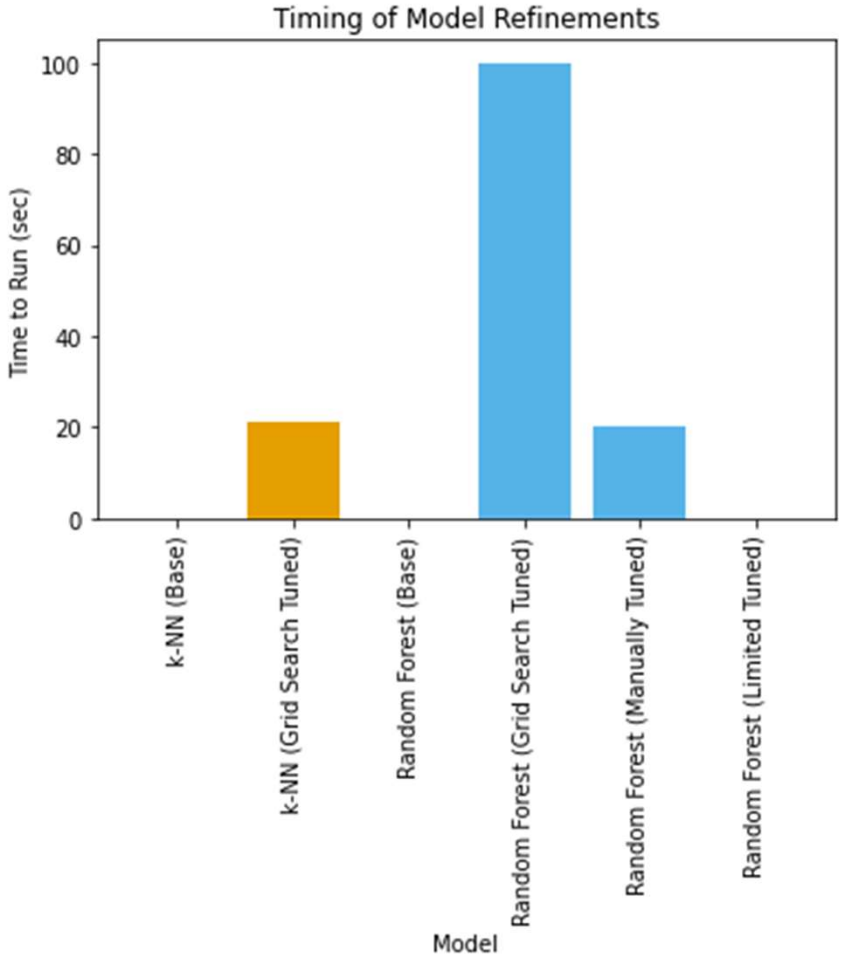


Limited



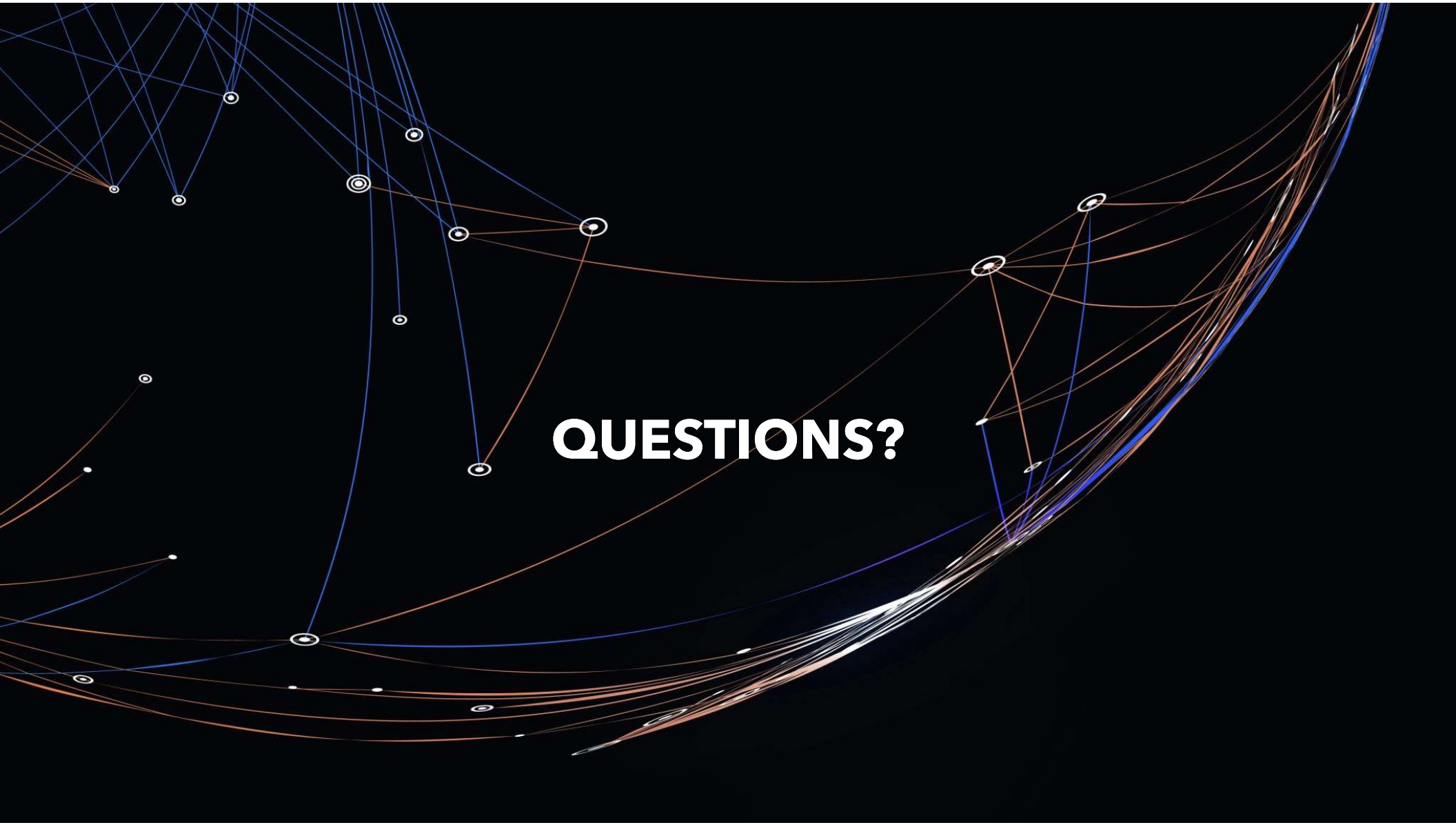
GridSearch often takes **too long** to be **practically useful**

Manual hyperparameter search dramatically **reduces computation time** with minimal accuracy loss



LIMITATIONS

- **Small** dataset
- **Older** data
  - Glass manufacturing evolves
- **Moderate variability**
  - Likely due to small dataset


QUESTIONS?