Classifying Microdebitage with Machine Learning

Presenters: Jose Cruz, Kimi Medina-Castellano,
Luke Taylor, Rodrigo Aguilar Barrios
Under mentorship of Dr. Phyllis Johnson and Dr. Reyes-Centeno

NACME
NATIONAL ACTION COUNCIL FOR MINORITIES IN ENGINEERING

University of Kentucky

Google

The Team

Rodrigo Aguilar Barrios

UC Berkeley
B.S. Electrical Engineering and Computer Science

Luke Taylor

University of Kentucky
B.S Computer Science
Digital Media and Design

Kimi Medina-Castellano

University of Kentucky
B.S. Mathematics
Physics & CS

Jose Cruz

University of Kentucky
B.S. Computer Science & Mathematics

Overview

01. Background

02. Data Analysis

03. Modeling

04. Future Steps

05. Acknowledgements

What is Microdebitage?

Leftover pieces (usually 6 mm or smaller) stones pieces that are knocked off from making stone tool

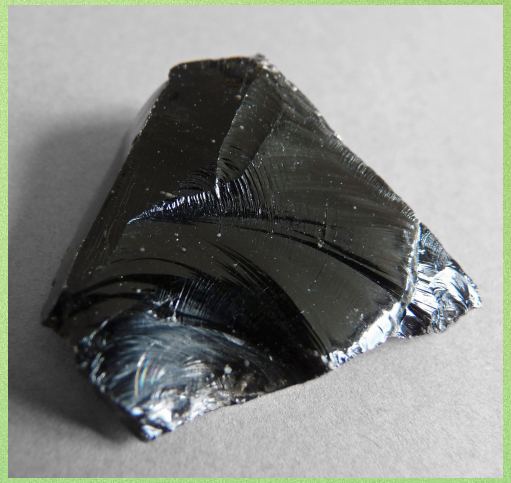
Tools have separate stages of production

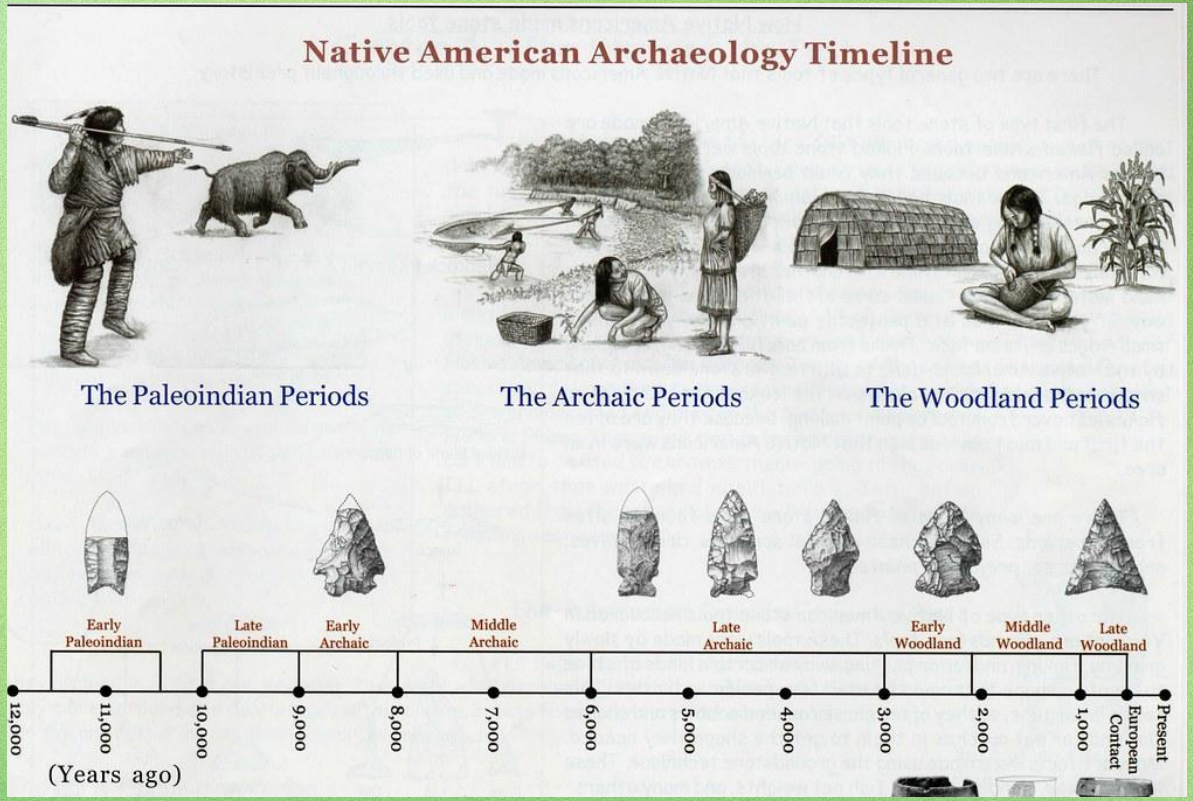
Requires precision and patience

Knapping

Tools can be recovered due to stone's resilience from human activity and natural processes

Chert and Obsidian




Native American Archaeology Timeline
The Paleoindian Periods
The Archaic Periods
The Woodland Periods
Early Paleoindian
Late Paleoindian
Early Archaic
Middle Archaic
Late Archaic
Early Woodland
Middle Woodland
Late Woodland
12,000
11,000
10,000
9,000
8,000
7,000
6,000
5,000
4,000
3,000
2,000
1,000
European Contact
Present
(Years ago)

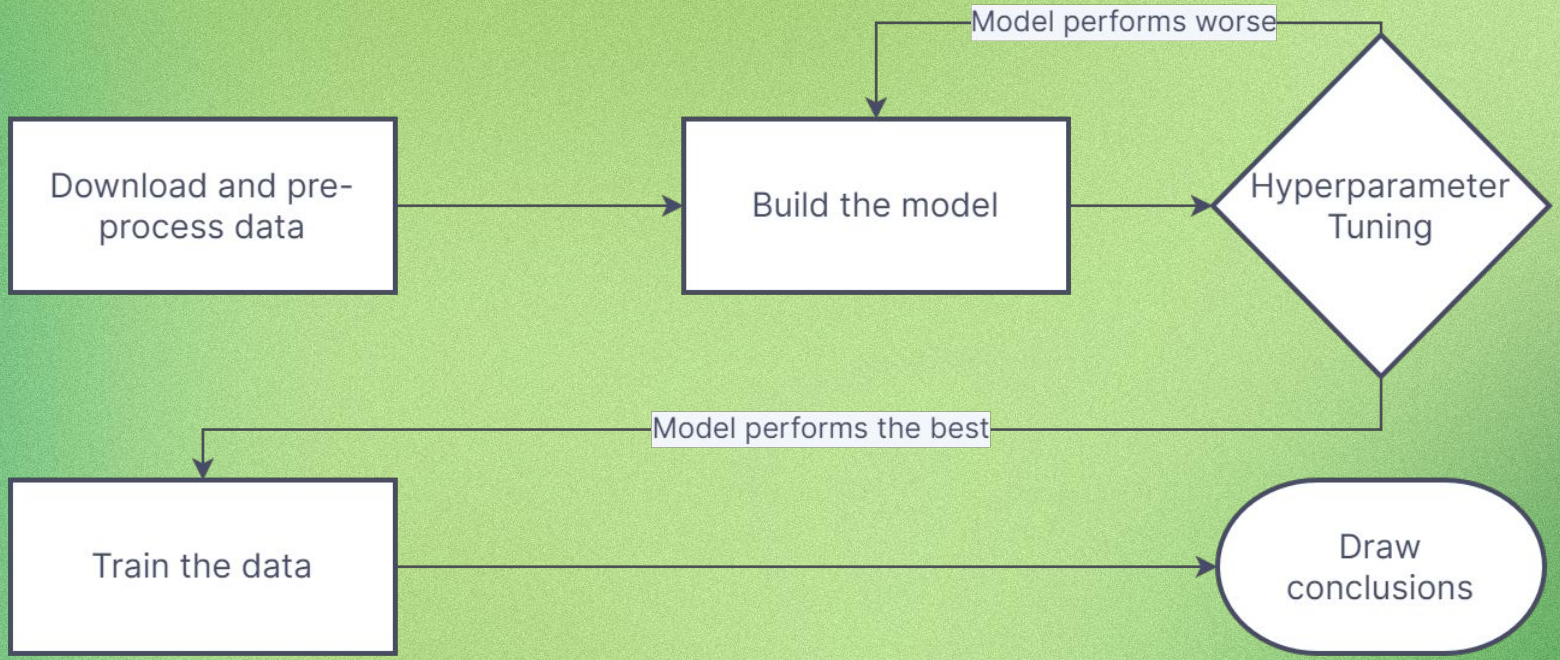
Problem to solve

Takes **>10 hours per sample** for researchers to visually analyze and sort the microdebitage

Solution: Create a model to classify microdebitage into certain stages of tool production

Why? To learn more about ancient civilizations

Procedure


Model performs worse
Download and pre-process data
Build the model
Hyperparameter Tuning
Model performs the best
Train the data
Draw conclusions

Acquiring Data

A professional knapper at Vanderbilt University makes simulants of stone debitage

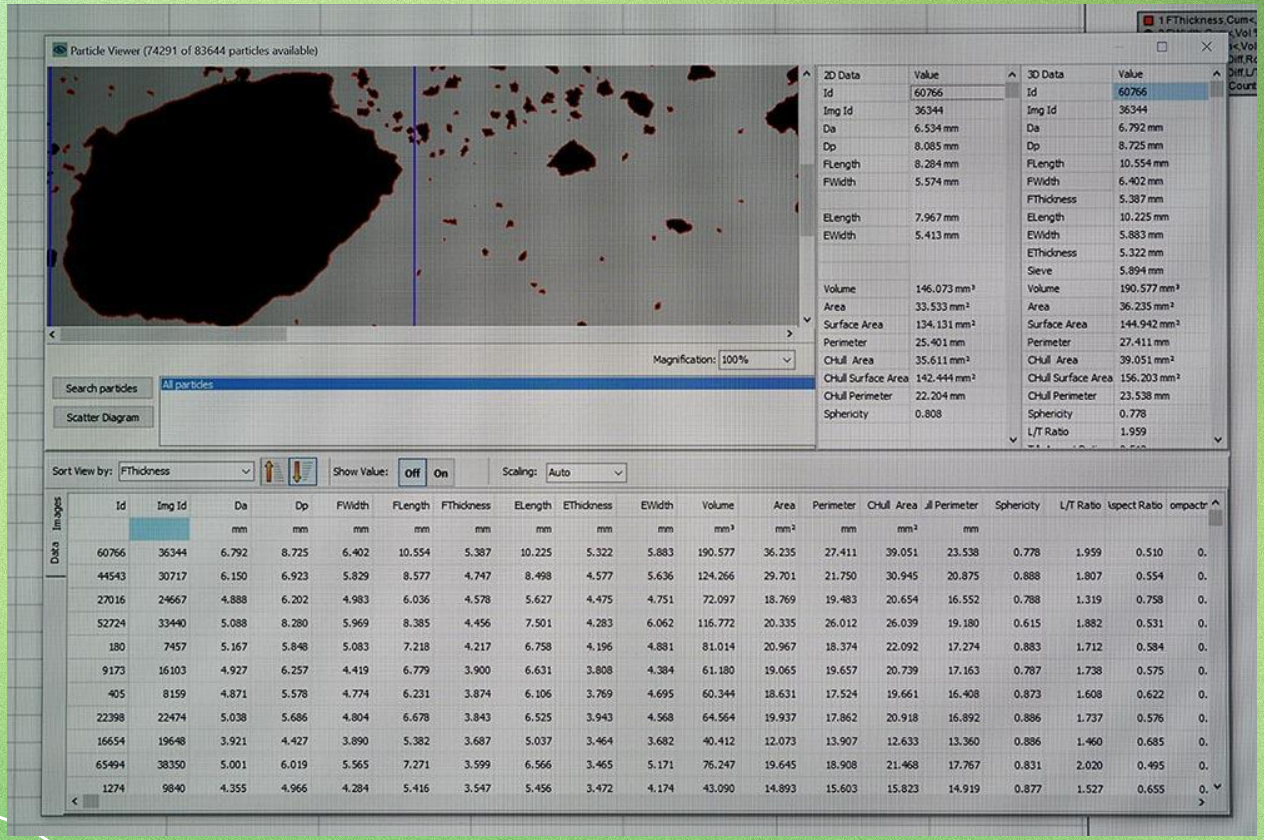
Knap

Microdebitage is gathered between stages

Gather Microdebitage

Sort through machine

Simulants are ran through Microtrac machine that takes images and numerical measurements of each particle


Particle Viewer (74291 of 83644 particles available)
Magnification: 100%
Search particles
Scatter Diagram
All particles
2D Data
Value
Id 60766
Img Id 36344
Da 6.534 mm
Dp 8.085 mm
FLength 8.284 mm
FWidth 5.574 mm
ELength 7.967 mm
EWidth 5.413 mm
Volume 146.073 mm³
Area 33.533 mm²
Surface Area 134.131 mm²
Perimeter 25.401 mm
CHull Area 35.611 mm²
CHull Surface Area 142.444 mm²
CHull Perimeter 22.204 mm
Sphericity 0.808
3D Data
Value
Id 60766
Img Id 36344
Da 6.792 mm
Dp 8.725 mm
FLength 10.554 mm
FWidth 6.402 mm
FThickness 5.387 mm
ELength 10.225 mm
EWidth 5.883 mm
EThickness 5.322 mm
Sieve 5.894 mm
Volume 190.577 mm³
Area 36.235 mm²
Surface Area 144.942 mm²
Perimeter 27.411 mm
CHull Area 39.051 mm²
CHull Surface Area 156.203 mm²
CHull Perimeter 23.538 mm
Sphericity 0.778
L/T Ratio 1.959
Sort View by: FThickness
Show Value: Off On
Scaling: Auto
Images
Data
Id Img Id Da Dp FWidth FLength FThickness ELength EThickness EWidth Volume Area Perimeter CHull Area ull Perimeter Sphericity L/T Ratio spect Ratio ompactr
mm mm mm mm mm mm mm mm mm³ mm² mm mm² mm
60766 36344 6.792 8.725 6.402 10.554 5.387 10.225 5.322 5.883 190.577 36.235 27.411 39.051 23.538 0.778 1.959 0.510 0.
44543 30717 6.150 6.923 5.829 8.577 4.747 8.498 4.577 5.636 124.266 29.701 21.750 30.945 20.875 0.888 1.807 0.554 0.
27016 24667 4.888 6.202 4.983 6.036 4.578 5.627 4.475 4.751 72.097 18.769 19.483 20.654 16.552 0.788 1.319 0.758 0.
52724 33440 5.088 8.280 5.969 8.385 4.456 7.501 4.283 6.062 116.772 20.335 26.012 26.039 19.180 0.615 1.882 0.531 0.
180 7457 5.167 5.848 5.083 7.218 4.217 6.758 4.196 4.881 81.014 20.967 18.374 22.092 17.274 0.883 1.712 0.584 0.
9173 16103 4.927 6.257 4.419 6.779 3.900 6.631 3.808 4.384 61.180 19.065 19.657 20.739 17.163 0.787 1.738 0.575 0.
405 8159 4.871 5.578 4.774 6.231 3.874 6.106 3.769 4.695 60.344 18.631 17.524 19.661 16.408 0.873 1.608 0.622 0.
22398 22474 5.038 5.686 4.804 6.678 3.843 6.525 3.943 4.568 64.564 19.937 17.862 20.918 16.892 0.886 1.737 0.576 0.
16654 19648 3.921 4.427 3.890 5.382 3.687 5.037 3.464 3.682 40.412 12.073 13.907 12.633 13.360 0.886 1.460 0.685 0.
65494 38350 5.001 6.019 5.565 7.271 3.599 6.566 3.465 5.171 76.247 19.645 18.908 21.468 17.767 0.831 2.020 0.495 0.
1274 9840 4.355 4.966 4.284 5.416 3.547 5.456 3.472 4.174 43.090 14.893 15.603 15.823 14.919 0.877 1.527 0.655 0.

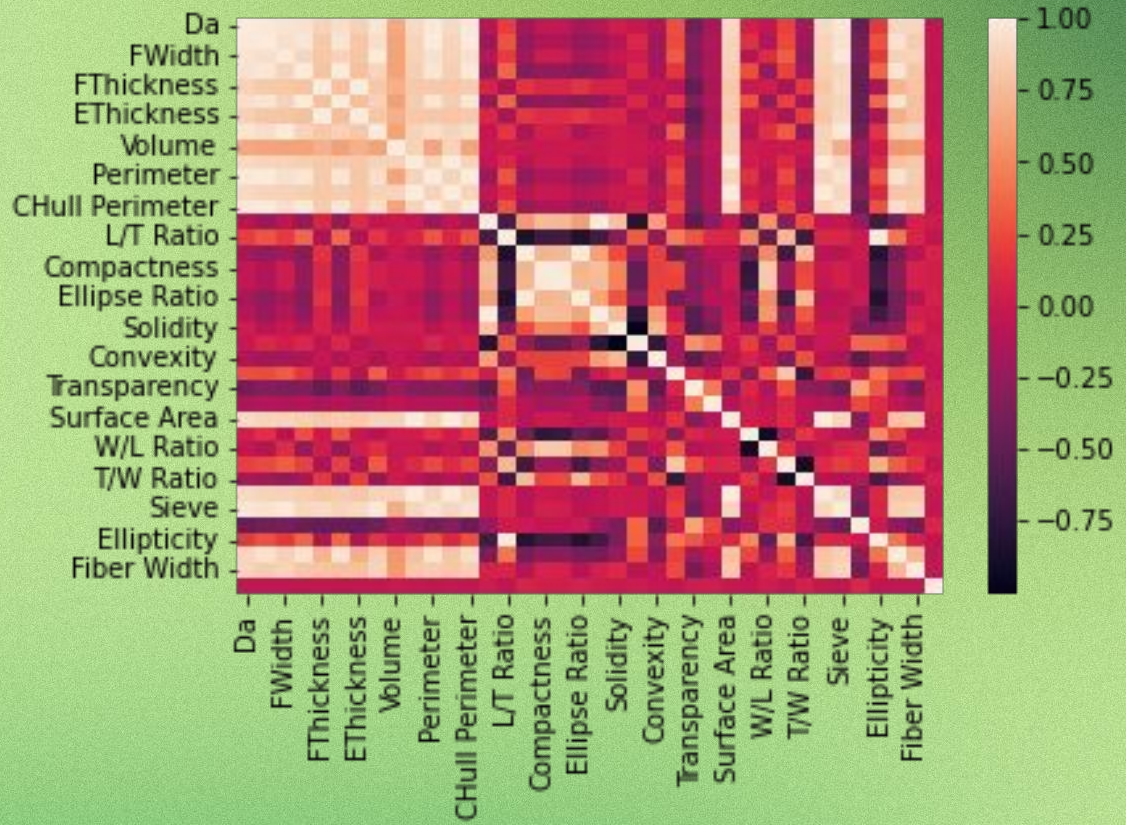
Preprocessing

The features that were taken out were the ones that had repeated values, such as the Filters and the hash columns. The Img Id and Id columns were deemed as not relevant to the model.

There were three columns that one of the data frames had, so we removed those columns from that particular dataframe

	Id	Filter0	Filter1	Filter2	Filter3	Filter4	Filter5	Filter6	hash	Img Id	Curvature	Transparency	Angularity
0	5903	Reject	Reject	Reject	Reject	Reject	Reject	Reject	0	14271	0	0.916	88.889
1	3002	Reject	Reject	Reject	Reject	Reject	Reject	Reject	0	11576	0.467	0.835	87.5
2	9893	Reject	Reject	Reject	Reject	Reject	Reject	Reject	0	18211	0.91	0.807	81.667
3	3713	Reject	Reject	Reject	Reject	Reject	Reject	Reject	0	12265	2.003	0.787	72.381
4	7862	Reject	Reject	Reject	Reject	Reject	Reject	Reject	0	16105	0	0.786	82.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...
36676	7	Reject	Reject	Reject	Reject	Reject	Reject	Reject	0	4022	0	0.154	18.989
36677	18	Reject	Reject	Reject	Reject	Reject	Reject	Reject	0	5347	0	0.154	21.2
36678	33	Reject	Reject	Reject	Reject	Reject	Reject	Reject	0	7645	0	0.153	18.144
36679	38	Reject	Reject	Reject	Reject	Reject	Reject	Reject	0	8347	0	0.148	17.347
36680	29	Reject	Reject	Reject	Reject	Reject	Reject	Reject	0	7032	0	0.144	29.632

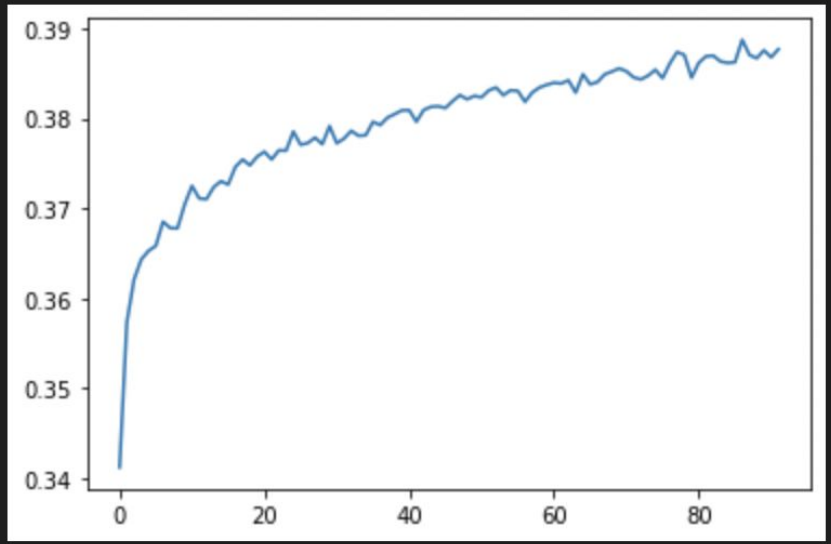
Data Visualization



Tensorflow Multiclass Classification

```
tf.keras.Sequential([
   tf.keras.layers.Dense(len(filtered) * 2,
input_shape=(len(filtered),)),
   tf.keras.layers.Dense(16, activation =
'relu'),
   tf.keras.layers.Dense(8, activation =
'relu'),
   tf.keras.layers.Dense(5, activation =
tf.nn.softmax)])

model.compile(
    loss='categorical_crossentropy',
    optimizer='Adam',
    metrics=['accuracy'],
)
```

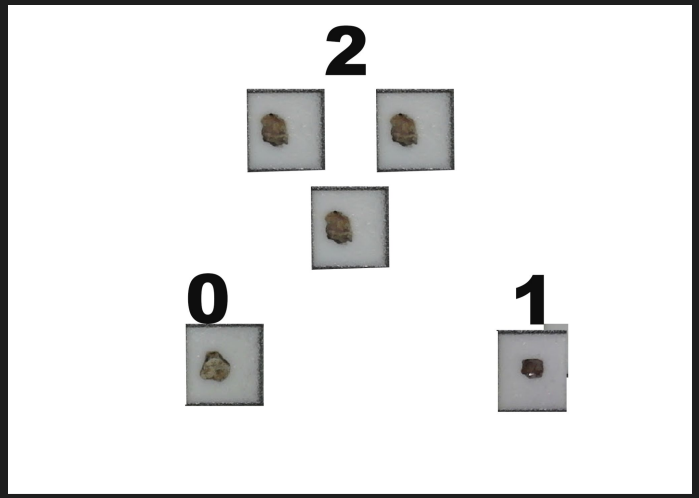


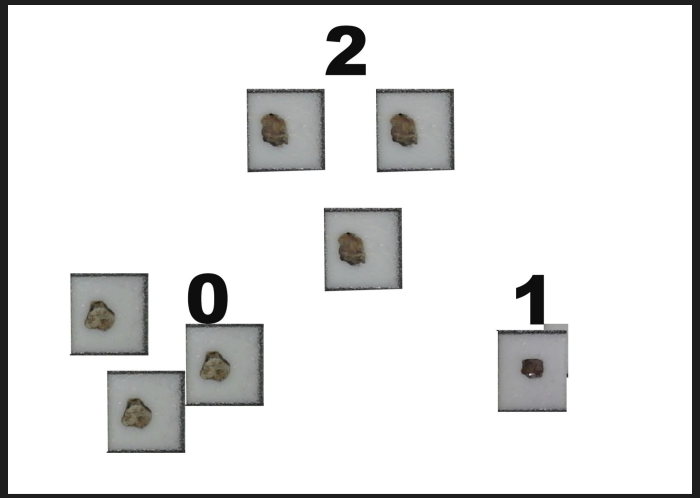
Graph of accuracy per epoch
loss: 1.4093 - accuracy: 0.3877
val_loss: 1.4085 - val_accuracy: 0.3884

KNeighborsClassifier

```
KNeighborsClassifier(n_neighbors=40, weights= 'distance', algorithm=  'brute', leaf_size = 35)
```

Accuracy: 0.3406

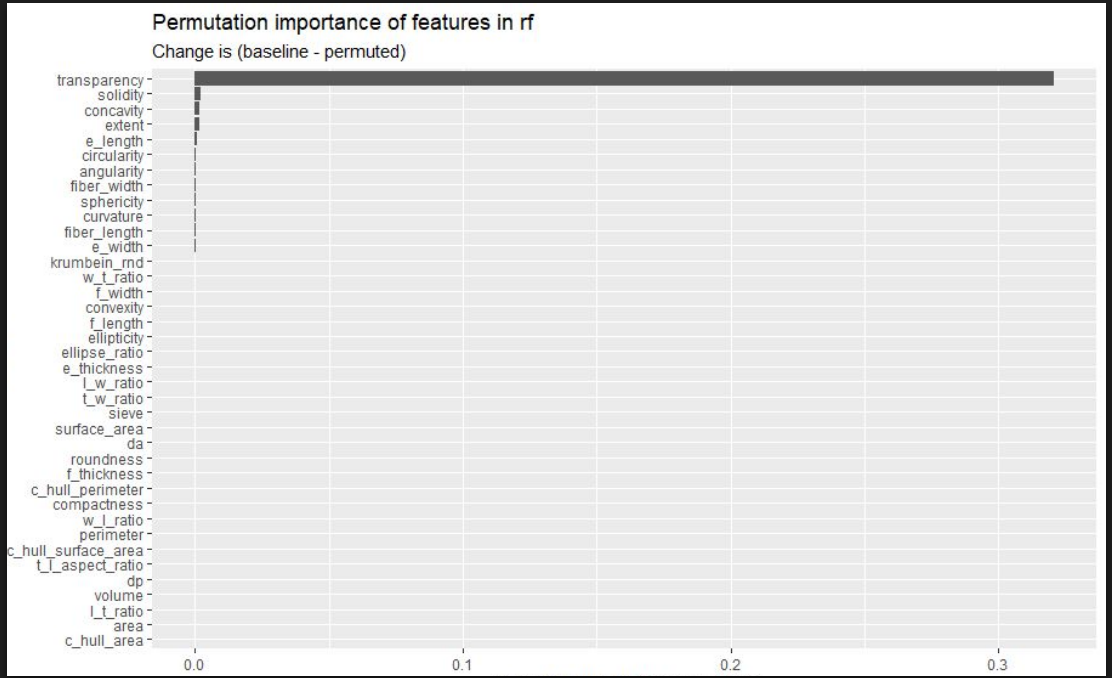




Importance of weights = distance

Making two models

	Angularity	Transparency	Curvature
0	88.889	0.916	0.0
1	87.5	0.835	0.467
2	81.667	0.807	0.91
3	72.381	0.787	2.003
4	82.5	0.786	0.0
...	...	...	...
49489	17.639	0.195	0
49490	25.781	0.306	0.184
49491	39.063	0.232	0
49492	48.889	0.389	0.37
49493	48	0.39	0.168



Two Model K-Neighbor Classifier

Modeling

```
O_model = KNeighborsClassifier(n_neighbors=5,
                               weights= 'distance',
                               p = 1,
                               leaf_size= 5,
                               algorithm='auto')
O_model.fit(O_data[O_filtered], O_data["Production Stage"])
labels = O_data['Production Stage']

random = O_data.sample(frac = 1)

one_hundred = random[:10000]

O_predictions = O_model.predict(X_test)

print('Accuracy score:',accuracy_score(y_test, O_predictions))
```

```
Accuracy score: 0.7373737373737373
```

```
C_model = KNeighborsClassifier(n_neighbors=5,
                               weights= 'distance',
                               p = 1, leaf_size= 5,
                               algorithm='auto')

C_model.fit(C_data[C_filtered], C_data["Production Stage"])
labels = C_data['Production Stage']

random = C_data.sample(frac = 1)

one_hundred = random[:10000]

C_predictions = C_model.predict(CX_test)

print('Accuracy score:',accuracy_score(cy_test, C_predictions))
```

```
Accuracy score: 0.4396887159533074
```

Decision Tree for Obsidian Model

Modeling

```
#Creating a list of the features and target
FEATURES = df.columns[:-1]

TARGET = df.columns[-1]


#Spliting our data for training and testing
x_train, x_test, y_train, y_test = train_test_split(
    df[FEATURES],
    df[TARGET],
    test_size = 0.2,
    random_state = 10
)
```

```
Accuracy:  0.8505909687847257
```

```
#Creating a decision tree object
dt = tree.DecisionTreeClassifier(
    criterion = 'entropy',
    max_depth = 5,
    random_state = 10
)

#Training our data
dt.fit(x_train, y_train)

#Making Predictions
prediction = dt.predict(x_test)

#Calculating our accuracy
acc = metrics.accuracy_score(y_test, prediction)
print('Accuracy: ', acc)
```

Modeling

Decision Tree: Chert

01.
```
# features and target
features = chert_df.loc[:, chert_df.columns != 'Stage']
feature_names = features.columns

target = chert_df['Stage']
target_name = 'Stage'
```

02.
```
# split data
x_train, x_test, y_train, y_test = train_test_split(
    features,
    target,
    test_size = 0.3,
    random_state = 10
)
```

03.
```
# train model
decision_tree = DecisionTreeClassifier()
decision_tree.fit(x_train, y_train)

dt = tree.DecisionTreeClassifier(
    criterion = 'gini'
)

dt.fit(x_train, y_train)
y_pred = dt.predict(x_test)
acc = metrics.accuracy_score(y_test, y_pred)
print('Accuracy: ', acc)
```

The chert model contained 3 datasets with 3 unique stages.

Accuracy: 43.75%

Will be tuning this model to increase accuracy

Medium accuracy can be due to missing columns

colab

TensorFlow

pandas

Obstacles

We were missing three crucial columns for some of our training datasets forcing us to get rid of those columns across all the data.

Running GridSearch to tune models on all the models took a very long time

Angularity	Transparency	Curvature
88.889	0.916	0
87.5	0.835	0.467
81.667	0.807	0.91
72.381	0.787	2.003
...	...	...
18.989	0.154	0
21.2	0.154	0
18.144	0.153	0
17.347	0.148	0
29.632	0.144	0

Ethical Consideration

- Misclassifying the time period
- Incorrect assumptions about time period
- Only looking at two materials
- Obtaining microdebitage

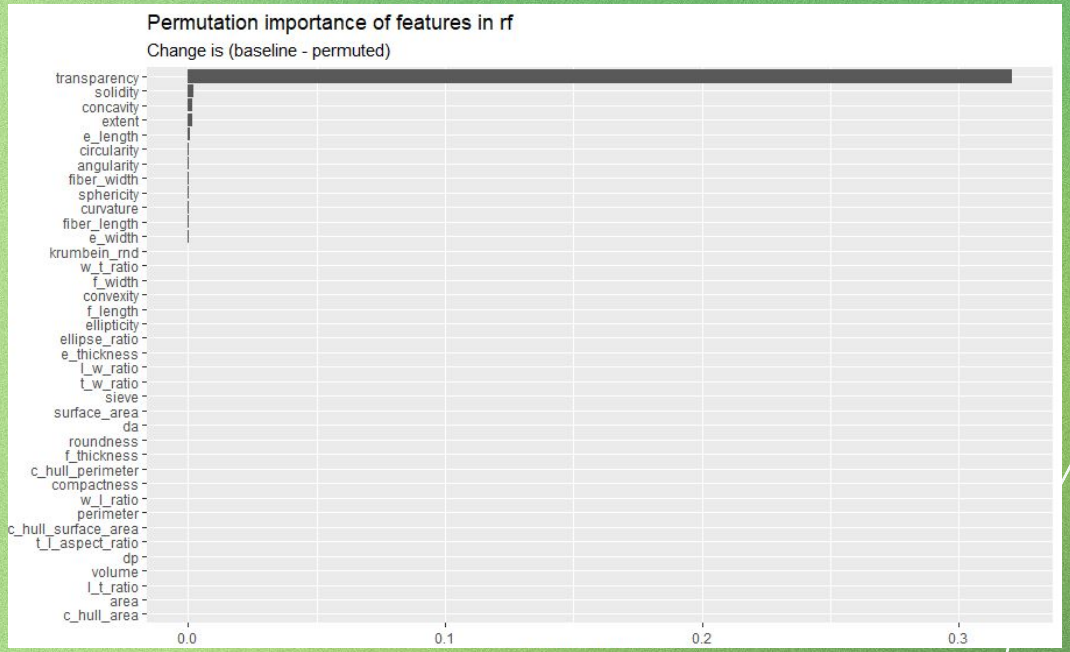
Moving Forward

Combining the chert model with the obsidian model

Combining the model that separates the soil and microdebitage with ours.

Re-analyzing the datasets that were missing columns to fill those data points and add those features into our training for an even better result.

Continuing research with mentors.



Acknowledgements

Dr. Corey Baker

Dr. Phyllis Johnson and mentors

University of Kentucky

NACME

Oscar Feliz

Jalen Collins

References

- https://towardsdatascience.com/how-to-tune-a-decision-tree-f03721801680
- https://scikit-learn.org/stable/
- https://www.xoriant.com/blog/decision-trees-for-classification-a-machine-learning-algorithm
- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
- Eberl, M., Johnson, P. , & Estrada Aguila, R. (2022). Studying lithic microdebitage with a dynamic image particle analyzer. *North American Archaeologist, 1-16.* DOI: 10.1177/01976931221109301
- Unpublished articles provided by Dr. Phyllis Johnson

Questions?