Feature Extraction

Data Import

```
ds = tabularTextDatastore("Files\*.txt");
```

Pre-Processing & Feature Extraction

```
preDS = transform (ds, @preprocess);
letterDS = transform(preDS, @extractfeatures);
```

Adding Character to data

```
% example row of processed data * number of files (each in this form)
letterSingleData = read(letterDS)
```

letterSingleData = 1×25 table

	AspectRatio	MADX	MADY	AvgU	MADU	AvgV	MADV	CorrXY
1	1.5667	0.1555	0.3363	0.3763	1.3595	-0.8364	3.5430	-0.1439

```
% all single row data combined to single table
letterdata = readall(letterDS);

% extracting character from files names
letterdata.Character = categorical(extractBetween(ds.Files,"_","_"))
```

letterdata = 3874×26 table

	AspectRatio	MADX	MADY	AvgU	MADU	AvgV	MADV	CorrXY
1	1.5667	0.1555	0.3363	0.3763	1.3595	-0.8364	3.5430	-0.1439
2	3.6790	0.1515	0.5988	0.1158	0.8987	-1.7412	3.1736	-0.4151
3	1.8667	0.2237	0.5152	0.0775	1.7327	-1.3236	1.6880	0.1900
4	2.0476	0.1838	0.3129	0.3508	1.2541	-0.3714	3.5946	0.5130
5	1.4054	0.1972	0.3153	0.6788	1.9449	-0.5584	2.3583	-0.1112
6	2.7742	0.1314	0.4757	0.4260	1.2890	-1.9871	3.3151	0.4028
7	2.5778	0.1683	0.4346	-0.8368	1.9298	-1.4155	2.4772	0.2995
8	4.4545	0.1495	0.6540	0.5167	0.4824	-2.3041	3.2084	-0.5370
9	5.8974	0.0974	0.7344	0.2830	0.9419	-1.4559	3.1839	-0.2079
10	3.8205	0.1069	0.5892	-0.7997	0.8569	-3.1871	3.4191	0.6417
11	3.3535	0.1180	0.5131	0.2768	1.2264	-1.7790	3.3186	0.1448
12	5	0.1456	0.6333	0.3856	0.7475	-0.5764	4.2179	0.6799
13	1.1064	0.2806	0.2562	0.9937	0.8669	-0.5991	3.4975	0.0663

	AspectRatio	MADX	MADY	AvgU	MADU	AvgV	MADV	CorrXY
14	1.6552	0.2706	0.3699	0.7371	0.7163	-1.5606	3.8920	0.1291

⋮

Divide data into sets for Training and Testing

```
% Random number state, for reproducibility
rng(123)

% Holdout partition to get indices for training and testing
partition = cvpartition(height(letterdata), "HoldOut", 0.25)
```

```
partition =
Hold-out cross validation partition
   NumObservations: 3874
       NumTestSets: 1
         TrainSize: 2906
          TestSize: 968
```

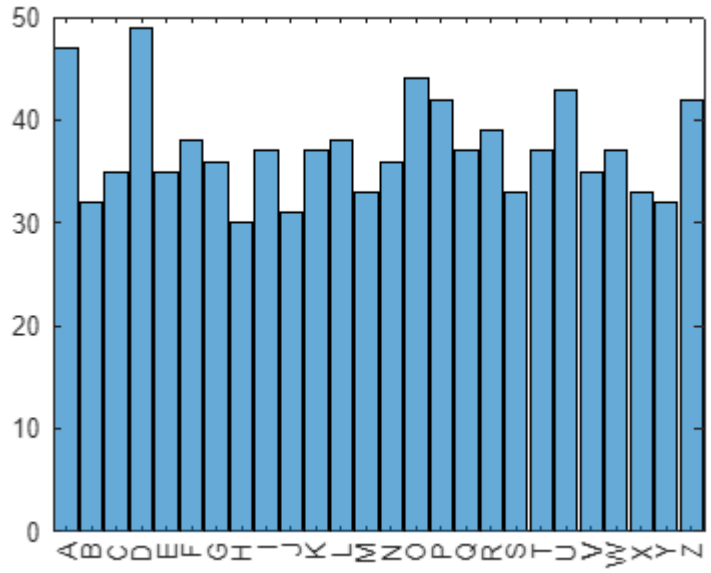
```
indTest = test(partition);
indTrain = training(partition);

% Indexing into data
testdata = letterdata(indTest, :);
traindata = letterdata(indTrain, :);
```

Record of the raw data files corresponding to the training and testing data

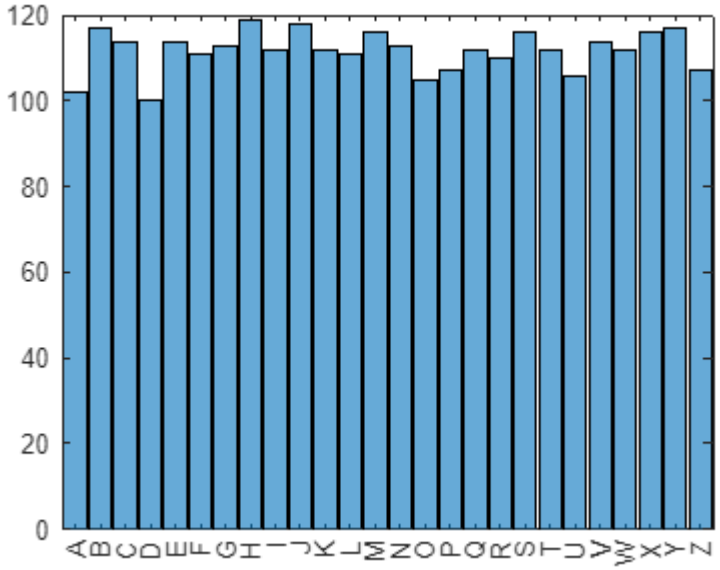
```
% sub-string extraction
files = extractAfter(ds.Files, "Files\");

testfiles = files(indTest);
trainfiles = files(indTrain);
```

Distribution

```
histogram(testdata.Character)
```




```
histogram(traindata.Character)
```



Data Writing

```
%save letterfeatures testdata traindata testfiles trainfiles
% wrtiting the complete data to single excel file
writetable(letterdata, "Data\letterComplete.xlsx");

% writing train and test data to their respective files
writetable(traindata, "Data\traindata.xlsx");
writetable(testdata, "Data\testdata.xlsx");

% writing test and train file names
```

```
writecell(testfiles, "Data\testFiles.xlsx");
writecell(trainfiles, "Data\trainFiles.xlsx");
```

Functions

function applied on each file and formated to table format*

*single row table, later combined to single table

Preprocessing

```
function data = preprocess(data)

    % Normalize time
    data.Time = (data.Time - data.Time(1)) / (data.Time(end) - data.Time(1));

    % Fixing aspect ratio to that of screen of writing device
    data.X = 1.5 * data.X;

    % Center X & Y at (0,0)
    data.X = data.X - mean(data.X, "omitnan");
    data.Y = data.Y - mean(data.Y, "omitnan");

    % Scale to have bounding box area of 1
    scale = 1 / sqrt(range(data.X) * range(data.Y));
    data.X = scale * data.X;
    data.Y = scale * data.Y;

    % Calculate derivatives
    data.dXdT = [NaN;diff(data.X)./diff(data.Time)];
    data.dYdT = [NaN;diff(data.Y)./diff(data.Time)];

    % Infinite value -> same values in data.Time
    data.dXdT(isinf(data.dXdT)) = NaN;
    data.dYdT(isinf(data.dYdT)) = NaN;

    % Smooth derivatives (moving average)
    n = round(0.1 * numel(data.Time));
    data.dXdT = movmean(data.dXdT,n);
    data.dYdT = movmean(data.dYdT,n);

end
```

Feature Extraction

```
function feat = extractfeatures(letter)

    % Correlations
    c = corrcoef (letter{:,2:end}, "Rows", "pairwise");

    % Parameters for finding local mins & maxes
```

```
    mp = 0.1;
    derivscale = 2;

    % Calculate features
    feat = [range(letter.Y) / range(letter.X),...                              %
aspect ratio
        mad(letter.X), mad(letter.Y),...                                       %
deviation for X & Y
        mean(letter.dXdT, "omitnan"), mad(letter.dXdT),...                     % mean
and deviation for X' & Y'
        mean(letter.dYdT, "omitnan"), mad(letter.dYdT),...
        c(1, 2:end), c(2, 3:end), c(3, 4:end), c(4, end),...                   %
correlations
        nnz(islocalmin(letter.X, "MinProminence", mp)),...                     %
number of local min/max for X & Y
        nnz(islocalmax(letter.X, "MinProminence", mp)),...
        nnz(islocalmin(letter.Y, "MinProminence", mp)),...
        nnz(islocalmax(letter.Y, "MinProminence", mp)),...
        nnz(islocalmin(letter.dXdT, "MinProminence", derivscale*mp)),...       %
number of local min/max for X' & Y'
        nnz(islocalmax(letter.dXdT, "MinProminence", derivscale*mp)),...
        nnz(islocalmin(letter.dYdT, "MinProminence", derivscale*mp)),...
        nnz(islocalmax(letter.dYdT, "MinProminence", derivscale*mp))];

    % Combine Features to single row table
    feat = array2table(feat,"VariableNames",...
        ["AspectRatio",...
        "MADX","MADY","AvgU","MADU","AvgV","MADV",...

"CorrXY","CorrXP","CorrXU","CorrXV","CorrYP","CorrYU","CorrYV","CorrPU","CorrPV","Co
rrUV",...
        "NumXMin","NumXMax", ...
        "NumYMin","NumYMax",...
        "NumUMin","NumUMax", ...
        "NumVMin","NumVMax"]);

end
```