**INFO204 Practical Test 1 Cheat Sheet**

**Reading CSV Files** - pd.read\_csv(‘file\_name.csv’)

**Merging CSV File –** df = df1.merge(df2, how = ‘inner’, on=’commonColumn’)

**Converting from Long to Wide –** df = df.pivot(index= ‘columnsForIndex’, columns=’measurementNames’, values=’measurementValues’).reset\_index()

**Convert from Wide to Long -** df.melt(var\_name='method', value\_name='MSE')

**Dropping Columns –** df.drop(columns=’column’, inplace= True)

**Saving to CSV File –** df.to\_csv(‘csv\_file\_name.csv’, index= False)

**Correlation Heatmap -**  sns.heatmap(df.corr())

**Pairwise Scatterplot –** sns.pairplot(df, aspect = 1, height = 3, corner = True)

**Extracting Columns into X and t –**

Target = ‘colname’

X = df.drop(columns=[target]).to\_numpy()

T = df[target].to\_numpy()

Feature\_names = df.drop(columns=target).columns.values

**Pipelines/Tuning Grids -**

Mlpipe = Pipeline([

(‘prepocess’, ‘passthrough’),

(‘model’, ModelName())

])

CART\_param\_grid = {

‘preprocess’ : [ ‘passthrough’],

‘model’ : [CART],

‘model\_\_min\_samples\_split’ : [minsamples]

}

For knn - \_\_n\_neighbors

For cart - \_\_min\_samples\_split

**np.logspace =**

To get 8 values between 2 and 256 evenly distributed and each value \* by 2

np.logspace(np.log2(2), np.log2(256), 8, base=2).astype(int)

**lineplots -** sns.lineplot(data=cv, x='min\_samples\_split', y='score', color='black', ax=axs[0]).set(title='CART', xscale='log', xlabel='$minsplit$')

sns.lineplot(data=cv\_stats, x='n\_neighbors', y='score', hue='weights', style='preprocess', ax=axs[1]).set(title='$k$-NN', xscale='log', xlabel='$k$ (Neighbourhood Size)')

**fig size -** plt.figure(figsize=20,16)

**Cross Validation on Tuning Grids –**

param\_grid[

CART\_param\_grid,

Knn\_param\_grid

]

Cv = GridSearchCV(mlpipe, param\_grid, cv = rkf)

cv.fit (X,t)

best\_model = cv.best\_estimator\_

print(cv.best\_params\_)

print(cv.best\_score\_)

**Cross Validation -** lm\_scores = cross\_val\_score(LinearRegression(), X, t, cv=rkf, scoring=mse\_score)

best\_scores = cross\_val\_score(best\_model, X, t, cv=rkf, scoring=mse\_score)

**make scorer** – mse\_score = make\_scorer(mean\_squared\_error)

**create dataframe –**

results = pd.DataFrame({

'Best from Grid Search' : best\_scores,

'Linear Regression' : lm\_scores

})

**Aggregate Data Result (MSE value for each method)** - display(df.groupby('method')[['MSE']].mean().reset\_index())

**Box plot -** sns.boxplot(data=results, x='method', y='MSE', ax=ax)

ax.set\_ylabel('MSE');

**Example Reasoning for Column Drops -** a and c appear to have little to no relationship with the response. While d and e appear to be uncorrelated to the response, they have a clear underlying non-linear relationship to the response (and so a non-linear learner may be able to exploit them). Finally, f has MANY missing values, so rather than remove instances with missing values, we will remove this column. (Lower number on the corr heatmap means less relationship)

**Example Scoring Reasoning -** Metric is MSE, so we aim to minimise. Linear regression can be considered a reasonable baseline of expected performance, and we can see that the model discovered through cross validation (standardised kNN, with a neighbourhood size of 4 and using distance weighting) is a substantial improvement over this.\_

**Recall –** aka Sensitivity is the fraction o f relevant instances that were retrieved, want to be higher. Evaluates row perf.

**Precision –** Fraction of relevant instances among the retrieved instances, evaluates Col perf.

**Accuracy -**  % of instances correctly predicted

**F1 Score -**  Balances recall and precision scores