**15 Sept 2018 – Linear Regression**

NLP –

Corpus - word2vec

Images are pixels – each pixel has R, B and G levels – 3 matrices overlayed on each other

Get\_dummies – one-hot encoding

Ordinal categories (grades A,B,C etc) might not need get\_dummies

**Interpreting the intercept (β0):**

* It is the value of y when all independent variables are 0.
* Here, it is the estimated number of rentals when the temperature is 0 degrees Celsius.
* **Note:** It does not always make sense to interpret the intercept. (Why?)

**Interpreting the "temp" coefficient (β1):**

* **Interpretation:** An increase of 1 degree Celcius is *associated with* increasing the number of total rentals by β1.
* Here, a temperature increase of 1 degree Celsius is *associated with* a rental increase of 9.17 bikes.
* This is not a statement of causation.
* β1 would be **negative** if an increase in temperature was associated with a decrease in total rentals.
* β1 would be **zero** if temperature is not associated with total rentals.

sklearn.preprocessing.StandardScalar

**22 Sept TT Split + Regularisation**

<https://codingstartups.com/practical-machine-learning-ridge-regression-vs-lasso/>

<https://chrisalbon.com/machine_learning/linear_regression/selecting_best_alpha_value_in_ridge_regression/>

Ridge vs Lasso (Feature selection)

Alpha – penalty term

Ridge only approaches zero

Elastic net applies both – balanced by two lambda parameters

Standardising predictors – should always implement!

print(kobe.columns[0:20]) – find columns

\*\*\* linreg = LinearRegression()

linreg\_scores = cross\_val\_score(linreg, Xs, y, cv=10)

print(linreg\_scores)

print(np.mean(linreg\_scores)) \*\*\*

cross val score does fitting for you

\*\*\*ridge\_alphas = np.logspace(0, 5, 200)

optimal\_ridge = RidgeCV(alphas=ridge\_alphas, cv=10)

optimal\_ridge.fit(Xs, y)

print(optimal\_ridge.alpha\_)\*\*\*

ridgecv does CVS for you.

Check KOBE SHOTS solutions again

**Train-test split**

* It's **low bias** because the models match the data effectively.
* It's **high variance** because the models are widely different, depending on which observations happen to be available in that universe. (For a body weight of 100 kg, the brain weight prediction would be 40 kg in one universe and 0 kg in the other!)

sns.lmplot() – plot the line + points

polynomials – start from low power and increase

fit on training

score/predict on test

score on training to test for overfitting

if overfitted, training score will be higher than test score

hyperparameters – gridsearch

scaling data:

MinMaxScaler – USECASE: can use if all columns are binary categories except one numerical column – apply to scale this kind of data

StandardScaler