Python - Linear Regression Model Cheat Sheet

```
# IMPORT DATA LIBRARIES import pandas as pd
import numpy as np # IMPORT VIS LIBRARIES
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# IMPORT MODELLING LIBRARIES
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearReg- ression
from sklearn import metrics
```

df = pd.read_csv('data.csv') read data df.head() check head df df.info() check info df df.describe() check stats df df.columns check col names

sns.pairplot(df) pairplot sns.distplot(df['Y']) distribution plot

sns.heatmap(df.corr(), annot=True) heatmap with values

CREATE X and y ---------------

X = df[['col1','col2',etc.]] create df features y = df['col'] create df var to predict **SPLIT DATASET** ---------------

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) split df in train and test df

FIT THE MODEL ---------------

lm = LinearRegression() instatiate model

lm.fit(X_train, y_train) train/fit the model **SHOW RESULTS** ---------------

lm.intercept_ show intercept

lm.coef_ show coefficients

coeff_df = pd.DataFrame (lm.coef_,X.columns,columns=['Coeff'])* create coeff df

pd.DataFrame: pd.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False). **data** = values, **index**= name index, **columns**= name column. This could be useful just to interpret the coefficient of the regression.

predictions = lm.predict(X_test) create predictions

plt.scatter(y_test,predictions)* plot predictions

sns.distplot((y_test-predictions),bins=50)* distplot of residuals

scatter: this graph show the difference between actual values and the values predicted by the model we trained. It should resemble as much as possible a **diagonal line**.

distplot: this graph shows the distributions of the residual errors, that is, the difference between the actual values minus the predicted values; it should result in an as much as possible **normal distribution**. If not, maybe change model!

print('MAE:', **metrics.mean_absolute_error(y_test, predictions)**)

print('MSE:', **metrics.mean_squared_error(y_test, predictions)**)

print('RMSE:', **np.sqrt(metrics.mean_squared_error(y_test, predictions))**)

MAE is the easiest to understand, because it's the average error. **MSE** is more popular than MAE, because MSE "punishes" larger errors, which tends to be useful in the real world.

RMSE is even more popular than MSE, because RMSE is interpretable in the "y" units.