Last updated 26/12/2020 **Created by Nabeel Khan**

To download: Toolbar > File > Download > (Desired File Type) https://github.com/nabeelkhan

☑	Stage	Steps	ADDITIONAL INFO	Useful Functions/Methods
☐	Prereq: Business & Data Understanding	Understand the data, your questions, and your goals • Are you simply exploring the data? • Are you preparing it for machine learning? • Is it in a tabular format? • How many features should I expect?	• Get a Data Dictionary or schema if possible • Understand what rows represent in your data • Studying the dataset for 1-2 hours will save you a ton of headache, especially if the dataset has >50 features	
☐	**I. Import Data & Libraries**	Download the data and make it available in your coding environment	• Import important libraries (pandas, numpy, matplotlib, seaborn, datetime), then import others as needed • Multiple datasets? Combine if you are concatenating (union). Otherwise, join when you understand them and are ready	• pd.concat • pd.merge
☐	**II. Exploratory Data Analysis**	Check for duplicates	• We don't need to keep any rows that are pure duplicates of each other	• df.drop_duplicates()
☐		Separate Data Types (Take an inventory of what data types you have)	• Numerical - Discrete - Continuous • Categorical - Ordinal - Nominal - Binary • Date/Time (time-stamps) • Text data (tweets/reviews) • Image • Sound	• df.select_dtypes(['object', 'bool']) • df.select_dtypes(['float', 'int']) • dtale.show() • df.info()
☐		Initial Data Cleaning • Clean anything that would prevent you from exploring the data	Examples of things to consider... • Are there categorical columns that should be numerical? • Is the data in the first few rows consistent with the name of the feature? • Are there lists or dictionaries packed into one feature? • Are dates in the date data type?	• pd.Series.str.replace() • pd.Series.astype() • pd.Series.map() • pd.Series.apply() • lambda functions • pd.cut() • sklearn.preprocessing.MultiLabelBinarizer • pd.to_datetime()
☐		Visualize & Understand • Understand how your data is distributed (numerical & categorical) • How are the columns related? (Find correlations or other relationships) • Are there any outliers? Note them (but don't remove them yet!) • This can also be a good time to do any statistical tests (T-tests maybe?) if you're interested	Some ideas • Numerical: Histograms & Scatter Plots • Categorical: Bar plots • Both: Box plots, violin plots, colored histograms • Date/Time: Line plots What data can tell you • Change Over Time • Hierarchy Drill Down • Zoom in and out of granularity • Contrasting Values • Intersections • Different Factors contributing to a larger phenomena • Outliers • Correlation	• df.value_counts() • seaborn.distplot() • seaborn.countplot() • matplotlib.pyplot.bar() • seaborn.FacetGrid() • df.groupby() • scipy.stats.ttest_ind()

✓	Stage	Steps	ADDITIONAL INFO	Useful Functions/Methods
☐		Assess Missing Values (**Don't fill/impute yet!**) • The goal here is to figure out your strategy for dealing with missing values since most ML algorithms cannot handle them. • You have 2 options: impute/fill them or remove them - For Imputing: skip below under **IV** for some imputation strategies - For Removing: try your best to critically think if removing is the best option for you ▫ Are there many missing values in one column? ▫ Are there many missing values in one row? ▫ Is a row missing the column you want to predict?	Things to consider when working with missing data... • How many per row? • How many per column? • Are they encoded as something else?	• df.isna().any() • df.drop() • np.isinf()
☐	**III. Train/Test Split**	Set aside some data for testing.	Depending on size of your data, this can be anywhere between 80-90% train.	• sklearn.model_selection.train_test_split • sklearn.model_selection.StratifiedShuffleSplit
☐	**IV. Prepare for ML**	Dealing with Missing Data (Many options) • Mean/Median/Mode • Find similar columns and fill • Fill with a unique value (like zero) • Predict Missing Values with ML - KNN (categorical) - Linear Regression (numerical) - Multiple Imputation or MICE for advanced methods - Maximum Likelihood Estimation	The reason we want to deal with missing data after we've split our data is because we want to simulate real world conditions when we test as much as we can. Some ideas: • Are there rows or columns you're okay with dropping? • Can you infer the value from other columns? • Categorical: most frequent may be a good option • Numerical: mean or median may be good options • See IterativeImputer for one method of using ML to fill multiple NA values - Key tradeoff between ML imputation and simple imputation... ▫ ML imputation gives you greater variability and precision in your features ▫ Simple imputation is much easier and less costly in production	• sklearn.impute.SimpleImputer • sklearn.impute.IterativeImputer • df.fillna() • fancyimpute.IterativeImputer
		Feature Engineering • What columns/features can you make to add value & information to your data?	Some ideas • Aggregations (across groups or dates) • Ratios (divide) • Interactions (multiply) • Frequency (counts) • Pull parts from dates (months/days/hours)	• sum • mean • / (divide) • df.groupby
☐		Transform Data • Numerical - Normalize or Standardize - Log-transform - Remove outliers • Categorical - One-hot encode (nominal) - Label encoder (ordinal) - Binarize (binary) • Text - Tokenize - Stem/Lemma - TF-IDF - (and much more NLP techniques)	Considerations: • Numerical - Some ML models perform better when features are all on the same scale - log-transforming can make numerical features seem more normal - removing outliers may increase your models' performance • Categorical - Try to avoid using pd.get_dummies if you want to replicate the transformation you fit during training onto your testing set - Use OneHotEncoder or other sklearn transformers instead	• sklearn.preprocessing.StandardScaler • sklearn.preprocessing.MinMaxScaler • sklearn.preprocessing.normalize • sklearn.preprocessing.LabelBinarizer • sklearn.preprocessing.MultiLabelBinarizer • sklearn.preprocessing.OneHotEncoder • pd.get_dummies • nltk.tokenize.word_tokenize • nltk.corpus.stopwords • nltk.stem.porter.PorterStemmer • nltk.stem.wordnet.WordNetLemmatizer • text.lower() • text.split() • sklearn.feature_extraction.text.CountVectorizer • sklearn.feature_extraction.text.TfidfVectorizer
☐		Feature Selection • Numerical: Correlation (Pearson or Spearman) or ANOVA • Categorical: Chi-Square test • Domain Knowledge • Recursive Feature Elimination (Like Forward Selection) • Low importance features (calculated via permutation_importance or feature_importance)	Reducing dimensionality of your data can not only improve runtime, but also the quality of your predictions. Highly correlated or low variance features might work against you. • Features you should consider removing... - Low variance (low variance = low information) - One of two highly correlated features (maybe corr > 0.95)? ▫ Pearson, Spearman, or ANOVA F-value - If categorical, high Chi-Squared statistic	• df.corr().abs() • sklearn.feature_selection.VarianceThreshold • sklearn.feature_selection.SelectKBest • sklearn.feature_selection.chi2 • sklearn.feature_selection.f_classif • sklearn.feature_selection.RFECV

☑	Stage	Steps	ADDITIONAL INFO	Useful Functions/Methods
☐	**V. Pick your Models**	• Some Regression Examples - Linear Regression - Support Vector Regressor - Random Forest - Boosted Trees - Neural Networks • Some Classification Examples - Support Vector Classifier - Random Forest - Logistic Regression - Boosted Trees - Neural Networks	**Go wild.**	
☐	**VI. Model Selection**	Pick one algorithm via some form of Cross-Validation	Cross validation is a great way to estimate how your models will perform out in the wild.	• sklearn.model_selection.train_test_split • sklearn.model_selection.KFold • sklearn.model_selection.StratifiedKFold • yellowbrick.classifier.roc_auc • yellowbrick.classifier.ClassificationReport • yellowbrick.regressor.ResidualsPlot
☐	**VII. Model Tuning**	Tune model hyperparameters • Ideally use Cross-Validation again to choose your hyperparameters	Some examples you can use • Grid Search • Random Search (Faster Grid Search) • Bayesian Optimization (Smarter Randomized Search) Also identify a good decision boundary (AKA discrimination threshold) if using classification • Can be done with Yellowbrick's quick DiscriminationThreshold viz	• sklearn.model_selection.GridSearchCV • sklearn.model_selection.RandomizedSearchCV • hyperopt library (Bayesian Optimization) • yellowbrick.classifier.DiscriminationThreshold
☐	**VIII. Pick the best model**	Pick the model that performed the best, and you're done!	**Woohoo!**	