

To download: Toolbar > File > Download > (Desired File Type)

<http://patrickdeguzman.me/>

<input checked="" type="checkbox"/>	Stage	Steps	ADDITIONAL INFO	Useful Functions/Methods
<input type="checkbox"/>	Prereq: Business & Data Understanding	<i>Understand the data, your questions, and your goals</i> <ul style="list-style-type: none"> 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? 	<ul style="list-style-type: none"> 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 	
<input type="checkbox"/>	I. Import Data & Libraries	Download the data and make it available in your coding environment	<ul style="list-style-type: none"> 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 	<ul style="list-style-type: none"> pd.concat pd.merge
<input type="checkbox"/>	II. Exploratory Data Analysis	Check for duplicates	<ul style="list-style-type: none"> We don't need to keep any rows that are pure duplicates of each other 	<ul style="list-style-type: none"> df.drop_duplicates()
<input type="checkbox"/>		Separate Data Types (Take an inventory of what data types you have)	<ul style="list-style-type: none"> Numerical <ul style="list-style-type: none"> - Discrete - Continuous Categorical <ul style="list-style-type: none"> - Ordinal - Nominal - Binary Date/Time (time-stamps) Text data (tweets/reviews) Image Sound 	<ul style="list-style-type: none"> df.select_dtypes(['object', 'bool']) df.select_dtypes(['float', 'int'])
<input type="checkbox"/>		Initial Data Cleaning <ul style="list-style-type: none"> Clean anything that would prevent you from exploring the data 	Examples of things to consider... <ul style="list-style-type: none"> 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? 	<ul style="list-style-type: none"> pd.Series.str.replace() pd.Series.astype() pd.Series.map() pd.Series.apply() lambda functions pd.cut() sklearn.preprocessing.MultiLabelBinarizer pd.to_datetime()
<input type="checkbox"/>		Visualize & Understand <ul style="list-style-type: none"> Understand how your data is distributed (numerical & categorical) How are the columns related? (Find correlations) 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: <ul style="list-style-type: none"> Numerical: Histograms & Scatter Plots Categorical: Bar plots Both: Box plots, violin plots, colored histograms Date/Time: Line plots 	<ul style="list-style-type: none"> df.value_counts() seaborn.distplot() seaborn.countplot() matplotlib.pyplot.bar() seaborn.FacetGrid() df.groupby() scipy.stats.test_ind()
<input type="checkbox"/>		Assess Missing Values (Don't fill/impute yet!) <ul style="list-style-type: none"> 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: <u>impute/fill</u> them or <u>remove</u> them - For <u>imputing</u>: skip below under IV for some imputation strategies - For <u>Removing</u>: try your best to critically think if removing is the best option for you <ul style="list-style-type: none"> 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... <ul style="list-style-type: none"> How many per row? How many per column? Are they encoded as something else? 	<ul style="list-style-type: none"> df.isna().any() df.drop() np.isinf()
<input type="checkbox"/>	III. Train/Test Split	Set aside some data for testing.	Depending on size of your data, this can be anywhere between 80-90% train.	<ul style="list-style-type: none"> sklearn.model_selection.train_test_split sklearn.model_selection.StratifiedShuffleSplit
<input type="checkbox"/>	IV. Prepare for ML	Dealing with Missing Data (Many options) <ul style="list-style-type: none"> Mean/Median/Mode Find similar columns and fill Fill with a unique value (like zero) Predict Missing Values with ML <ul style="list-style-type: none"> - KNN (categorical) - Linear Regression (numerical) 	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: <ul style="list-style-type: none"> 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 NA 	<ul style="list-style-type: none"> sklearn.impute.SimpleImputer sklearn.impute.IterativeImputer df.fillna()
<input type="checkbox"/>		Feature Engineering <ul style="list-style-type: none"> What columns/features can you make to add value & information to your data? 	Some ideas: <ul style="list-style-type: none"> Aggregations (across groups or dates) Ratios (divide) Interactions (multiply) Frequency (counts) Pull parts from dates (months/days/hours) 	<ul style="list-style-type: none"> sum mean / (divide) df.groupby
<input type="checkbox"/>		Transform Data <ul style="list-style-type: none"> Numerical <ul style="list-style-type: none"> - Normalize or Standardize - Log-transform - Remove outliers Categorical <ul style="list-style-type: none"> - One-hot encode (nominal) - Label encoder (ordinal) - Binarize (binary) Text <ul style="list-style-type: none"> - Tokenize - Stem/Lemma - TF-IDF - (and much more NLP techniques) 	Considerations: <ul style="list-style-type: none"> Numerical <ul style="list-style-type: none"> - 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 <ul style="list-style-type: none"> - 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 	<ul style="list-style-type: none"> 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.CountVecorizer sklearn.feature_extraction.text.TfidfVecorizer
<input type="checkbox"/>		Feature Selection <ul style="list-style-type: none"> 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... <ul style="list-style-type: none"> Low variance (low variance = low information) One of two highly correlated features (maybe corr > 0.95)? <ul style="list-style-type: none"> - Pearson, Spearman, or ANOVA F-value If categorical, high Chi-Squared statistic 	<ul style="list-style-type: none"> df.corr().abs() sklearn.feature_selection.VarianceThreshold sklearn.feature_selection.SelectKBest sklearn.feature_selection.chi2 sklearn.feature_selection.f_classif sklearn.feature_selection.RFECV
<input type="checkbox"/>	V. Pick your Models	<ul style="list-style-type: none"> Some Regression Examples <ul style="list-style-type: none"> - Linear Regression - Support Vector Regressor - Random Forest - Boosted Trees - Neural Networks Some Classification Examples <ul style="list-style-type: none"> - Support Vector Classifier - Random Forest - Logistic Regression - Boosted Trees - Neural Networks 	Go wild.	
<input type="checkbox"/>	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.	<ul style="list-style-type: none"> sklearn.model_selection.train_test_split sklearn.model_selection.KFold sklearn.model_selection.StratifiedKFold
<input type="checkbox"/>	VII. Model Tuning	Tune model hyperparameters <ul style="list-style-type: none"> Ideally use Cross-Validation again to choose your hyperparameters 	Some examples you can use <ul style="list-style-type: none"> Grid Search Random Search (Faster Grid Search) Bayesian Optimization (Smarter Randomized Search) 	<ul style="list-style-type: none"> sklearn.model_selection.GridSearchCV sklearn.model_selection.RandomizedSearchCV hyperopt library (Bayesian Optimization)
<input type="checkbox"/>	VIII. Pick the best model	Pick the model that performed the best, and you're done!	Woohoo!	