SI 370 Cheat Sheet: Six Steps to Data Science

Example code in this document uses the housing prices in San Francisco dataset. See GitHub Repo here.

1) Exploratory Data Analysis

Key Packages: pandas, numpy

Describing Data:

- .describe(): returns descriptive statistics of column, by default only looks at numeric columns
- Use pd.isnull() to check is a value is missing and .fillna() to impute a value into missing cells
- Aggregations: use .groupby() alongside an aggregation function (i.e. .max, .mean) or .agg()
  - ```
    df.groupby('bedrooms')['price'].agg(['max', 'mean'])
    ```
- Crosstab: Crosstabulaton of 2 variables. Defaults to frequency table, unless a cell aggregation is specified
  - ```
    pd.crosstab(index = df.bedrooms, columns = df.bathrooms, values = df.price, aggfunc = np.median)
    ```

Combining Data:

- Merge: More flexible than a join, allow us to combine data frames based on their columns or indexes
  - ```
    neighborhood_1.merge(neighborhood_2, on = 'date')
    ```
- Concat: Concentrates data frames along an axis (rows or columns), essentially "adds" the dataframes together
  - ```
    along_rows = pd.concat([neighborhood_1, neighborhood_2], axis = 0)
    ```

2) Data Visualization

Key Packages: seaborn, matplotlib

Visualization Tricks:

- Things to consider when plotting: using aggregation functions, visual hierarchy, scale transformations (i.e.using a log scale), adding annotations to the chart, labels and legend formatting
- Useful plot types: regplot (scatterplot + regression line), pairplot (shows relationship between all variable included), heatmap (co-occurrences of categories, use with crosstab)
- Plotting Example:
  - ```
    sns.set(rc={'figure.figsize':(18, 12)})
    f, axes = plt.subplots(1, 2)
    f1 = sns.histplot(x = df.price_bin, y = df.bedrooms, stat = 'density', multiple = 'dodge', ax = axes[0])
    for tick in axes[0].get_xticklabels():
        tick.set_rotation(45)
    _ = f1.set(title = 'Density Plot of Price and Number of Bedrooms', xlabel = 'Price Range', ylabel = 'Bedrooms')
    f2 = sns.regplot(data = df, x = 'price', y = 'square_feet', ax = axes[1])
    _ = f2.set(title = 'Regression of Priice vs Square Feet', xlabel = 'Price (USD)', ylabel = 'Square Feet')
    ```

3) Statistical Analysis

Key Packages: statsmodel, scipy.stats

Numeric Data:

- Regression: Model relationship between a response variable and one or more explanatory variables
  - .add_constant() adds the y-intercept
  - ```
    lm = statsmodels.formula.api.ols('price ~ bathrooms + bedrooms + square_feet + C(neighborhood)', data = df).fit()
    ```
- ANOVA: Analyze difference between groups→ F Statistic = Variation between groups / variation within group
  - ```
    anova_table = sm.stats.anova_lm(lm)
    ```
- Tukey's Honestly Significant Difference: Compares all possible pairs of group means to see if they're significantly different

Categorical Data:

- Chi-2 test: Tests if the counts in a contingency table are independent of each other
  - ```
    chi2, p_val, degrees_freedom, exp_val = scipy.stats.chi2_contingency(pd.crosstab(df.bedrooms, df.bathrooms))
    ```

4) Machine Learning: Pipelines, Dimension Reduction & Clustering
Key Packages: scikit-learn
Key Concepts:
- In supervised approaches, we have known labels we use to train our model. In unsupervised learning, algorithms find patterns in the dataset without the need for labels
- Pipelines help avoid leakage between the train and test data during preprocessing

Dimension Reduction:
- Multidimensional Scaling: Creates distance between data points in 2D that are similar to the distances in nD → visualization dimensions aren't directly related to original dimensions
    - Metric MDS: Entries in matrix represent a distance between items, use non-metric otherwise
    - ```
      mds = manifold.MDS(n_components = 2, metric = False, eps = 1e-9, random_state = 42, dissimilarity = 'euclidean', n_jobs = 1)
      mds = mds.fit_transform(X)
      ```
- t-SNE: Assigns points probability based on their similarity, plots similar points close and dissimilar points far in 2D space → non-convex, so different initialization gives different results, computationally intensive
- Principal Components Analysis: Projects k dimensional cloud of points onto a 2d surface, finds projection that captures the most variance → very sensitive to scaling

Clustering:
- k-Means: Divide into k clusters based on distance to a centroid/mean → sensitive to centroid initialization

5) Machine Learning: Classification
Key Packages: scikit-learn
Key Concepts:
- Try to predict label based on features in data → if you use lots of features you should have lots of data
- Need to divide data into train, test and (optionally) validation set
- Useful classifiers: Decision Trees, Naive Bayes, Support Vector Machines, K-Nearest Neighbors, etc.
- Grid Search Cross Validation is one method for findings the best performing hyperparameters
    - ```
      estimators = {'gamma': ['scale',  'auto'], 'decision_function_shape': ['ovo', 'ovr']}
      svc = SVC(random_state = 42)

      svc_model = GridSearchCV(svc, estimators, cv = 3, verbose = 3.1)
      svc_model.fit(X_train, y_train)
      ```
- A classification report will show precision, recall and the F1 score for each class
    - ```
      classification_report(svc_model.predict(X_test), y_test)
      ```

6) Natural Language Processing
Key Packages: scikit-learn, NLTK
Preprocessing:
- Normalize, remove stop words and lemmatize (optional) with NLTK. Examine text to see if other features should be removed (e.g. url's, hashtags, non-ASCII characters)
- Text needs to be converted to a numerical vector for it to be machine readable
    - Tf-IDF: # of times a word appears in document / # of documents it appears in → scale down weights of tokens that are less informative (i.e. appear in many documents), focus on informative tokens
    - ```
      tfidf_vectorizer = TfidfVectorizer(analyzer = 'word', max_features = 500)
      X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
      X_test_tfidf = tfidf_vectorizer.transform(X_test)
      ```
    - Countvectorizer: How often words appear in text, can be bag-of-words or n-gram counts