PyData 2016 sklearn Pipelines

August 28, 2016

1 Deploying Machine Learning using `Sklearn` `Pipelines`

1.1 Kevin Goetsch

1.1.1 GRUBHUB

2 What is this Talk about?

"Pipeline" is an overloaded phrase, every library, every paradigm uses the term.

I am going to talk about a specific technique, in a library, in a language, that you're using for machine learning - Machine Learning - Python - scikit-learn - `Pipeline`

3 Why are you watching this?

- Understand how and when to build `Pipelines`
- Learn how to smooth out your development and deployment of predictive models
- Because you're my father and you still don't know what I do at work

4 Road Map

- Sklearn Pipelines
- Modeling Example
- Piece by Piece walkthrough
- Check-in
- Data Science Modeling Lifecycle
  - Development
  - Deployment
- Wrap

5 What are Sklearn `Pipelines`?

- `Pipelines` are containers of steps
  - `Transformer`
  - `Estimator`

  - `Pipeline`
  - `FeatureUnion`

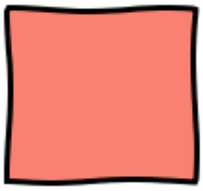
- `Pipelines` are used to package a workflow and fit model into a signle object

6 Building Blocks of Sklearn Pipelines

6.0.1 They're like Legos!

6.1 `Estimator`

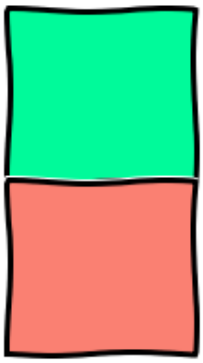
```
In [2]: estimator_visual()
```



6.2 `Transformer` & `Estimator`

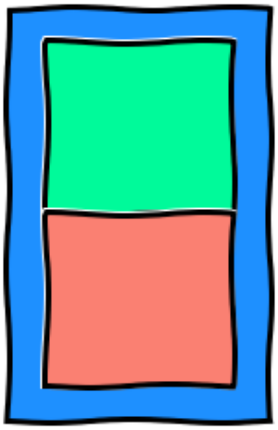
Note: No communication between the two

```
In [3]: transformer_visual()
```



6.3 `Pipeline` Routes Output of `Transformer` as input into `Estimator`

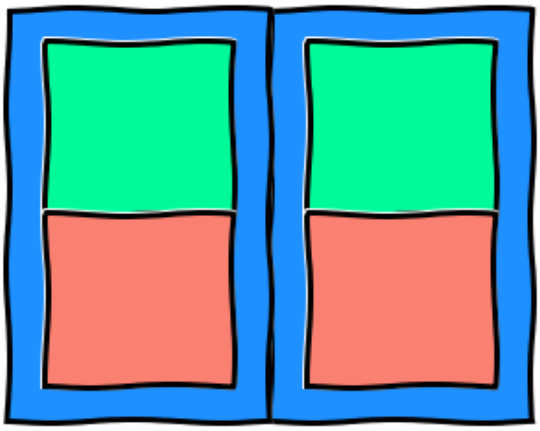
```
In [4]: pipeline_visual()
```



6.4 Two side by side `Pipelines`

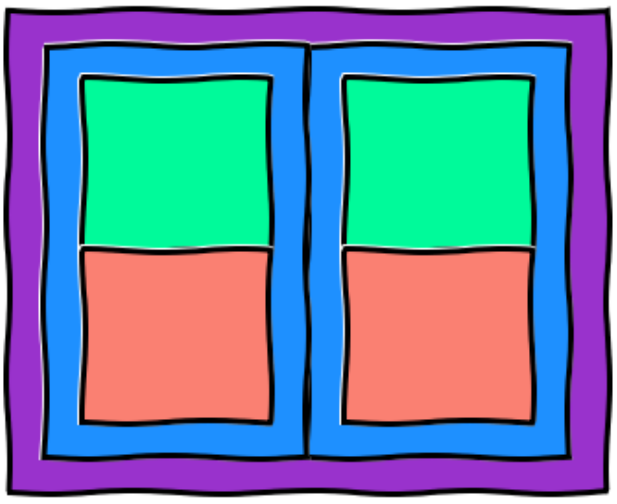
Note: No communication between the two

```
In [5]: two_pipelines_visual()
```



6.5 `FeatureUnion` joins the results of both `Pipelines` outputs

```
In [6]: featureUnion_visual()
```



7 Modeling Example

Titantic Dataset - predict survival

7.1 Non-Pipeline

```
In [7]: response_var = 'Survived'
        # TRAIN DATA
        train_df = pd.read_csv('data/train.csv', header=0)

        # All the ages with no data -> make the median of all Ages
        median_age = train_df['Age'].dropna().median()
        if len(train_df.Age[ train_df.Age.isnull() ]) > 0:
            train_df.loc[ (train_df.Age.isnull()), 'Age'] = median_age

        train_df = train_df.drop(['Pclass',
         'Name',
         'Sex',
         'SibSp',
         'Parch',
         'Ticket',
         'Fare',
         'Cabin',
         'Embarked'], axis=1)

In [8]: # TEST DATA
        test_df = pd.read_csv('data/test.csv', header=0)

        median_age = test_df['Age'].dropna().median()
        if len(test_df.Age[ test_df.Age.isnull() ]) > 0:
            test_df.loc[ (test_df.Age.isnull()), 'Age'] = median_age

        test_df = test_df.drop(['Pclass',
         'Name',
         'Sex',
         'SibSp',
         'Parch',
         'Ticket',
         'Fare',
         'Cabin',
         'Embarked'], axis=1)

In [9]: forest = RandomForestClassifier(n_estimators=100,random_state=2065)
        forest = forest.fit( train_df[['Age']], train_df[response_var] )

        forest_output = forest.predict_proba(test_df[['Age']])
```

7.2 Redone with a Pipeline

```
In [10]: response_var = 'Survived'
         train_df_pipe = pd.read_csv('data/train.csv', header=0)
         test_df_pipe = pd.read_csv('data/test.csv', header=0)

         pipeline = make_pipeline(FactorExtractor('Age'),
                                  FillNA(train_df_pipe['Age'].dropna().median()),
                                  ConvertToDataFrame(),
                                  RandomForestClassifier(n_estimators=100, random_s

In [11]: pipeline.fit(train_df_pipe, train_df_pipe[response_var].values )
         pipe_output = pipeline.predict_proba(test_df_pipe)
```

7.2.1 Pipeline's prediction should match our earlier work

```
In [12]: (pipe_output == forest_output).all()

Out[12]: False
```

7.3 What?

earlier...

```
median_age = test_df['Age'].dropna().median()

In [13]: test_df = pd.read_csv('data/test.csv', header=0)

In [14]: test_df['Age'].dropna().median() == train_df['Age'].dropna().median()

Out[14]: False

In [15]: test_df['Age'].dropna().median()

Out[15]: 27.0

In [16]: train_df['Age'].dropna().median()

Out[16]: 28.0
```

7.4 Rerun scoring with fixed Median Age

8 ...

```
In [17]: test_df = pd.read_csv('data/test.csv', header=0)

         median_age = train_df['Age'].dropna().median()
         if len(test_df.Age[ test_df.Age.isnull() ]) > 0:
             test_df.loc[ (test_df.Age.isnull()), 'Age'] = median_age
```

```
        test_df = test_df.drop(['Pclass',
         'Name',
         'Sex',
         'SibSp',
         'Parch',
         'Ticket',
         'Fare',
         'Cabin',
         'Embarked'], axis=1)
```

```
In [18]: forest = RandomForestClassifier(n_estimators=100,random_state=2065)
         forest = forest.fit( train_df[['Age']], train_df[response_var] )

         forest_output = forest.predict_proba(test_df[['Age']])
```

```
In [19]: (pipe_output == forest_output).all()
```

```
Out[19]: True
```

This is a common mistake that can be avoided by using pipelines

9 Modeling Wrap Up (or why `Pipelines`?)

- Transformations written out once
- Easy to swap out pieces
- Readability (no more cursing Past Kevin)
- Keeps all intermediate steps together

9.0.1 But what was really going on in all those steps?

10 What's a Transformer?

Python Class that has two main methods - `transform(X, [y])` - Applies transformations on X - `fit(X, [y])` - Applies fit logic on X

```
In [20]: class FactorExtractor(TransformerMixin, BaseEstimator):
             def __init__(self, factor):
                 self.factor = factor

             def transform(self, data):
                 return data[self.factor]

             def fit(self, *_):
                 return self
```

Calling `transform` on this Transformer is identical to manually subselecting the target factor.

```
In [21]: ( FactorExtractor('Age').transform(train_df) == train_df['Age'] ).all()
```

```
Out[21]: True
```

11 What's an Estimator

General term for sklearn objects which make predictions.

Methods we'll focus on: - `fit(X, y)` - Fits the model object on the data - `predict(X)` - Predict class labels for samples in X

```
In [22]: RandomForestClassifier(n_estimators=100,random_state=2065)

Out[22]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini'
                        max_depth=None, max_features='auto', max_leaf_nodes=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
                        oob_score=False, random_state=2065, verbose=0,
                        warm_start=False)
```

12 What's a Pipeline?

Pipeline of transforms with a final estimator - Transform(X, [y]) - Call `fit` then `transform` on each step contained inside - Predict(X) - Call `transform` on each step then `predict` on the final step

Initialized by passing a list of tuples in the form of

```
(name, transformer)
```

`make_pipeline` just shortcuts the creation process by automatically creating the tuples from a list of transformers

```
make_pipeline(FactorExtractor('Age'),
              FillNA(),
              ConvertToDataFrame(),
              RandomForestClassifier(n_estimators=100, random_state=2065))
```

Same things as . . .

```
Pipeline([
    ('factorextractor', FactorExtractor('Age')),
    ('fillna', FillNA()),
    ('converttodataframe', ConvertToDataFrame()),
    ('randomforestclassifier', RandomForestClassifier(random_state=2065))
    ]
   )
```

13 What's a FeatureUnion?

Basically a horizontal Pipeline - Transform(X, [y]) - Call `fit` then `transform` on each step contained inside

Merges the final result of each step into a single output

Initialized by passing a list of tuples in the form of

```
(name, transformer)
```

Caveat: I've written a `DataFrameUnion` class that is identical to `FeatureUnion` except it returns a Pandas dataframe

```
In [23]: train_df_pipe[['Age', 'Sex']].head(3)
```

```
Out[23]:     Age     Sex
         0  22.0    male
         1  38.0  female
         2  26.0  female
```

```
In [24]: feature_union = make_dataframeunion([
             make_pipeline(FactorExtractor('Age'),
                           FillNA()),
             make_pipeline(FactorExtractor('Sex'),
                           FillNA('missing'),
                           CategoricalDummifier())
             ])
```

```
In [25]: feature_union.fit_transform(train_df_pipe).head(3)
```

```
Out[25]:   Age  Sex[T.male]
         0  22          1.0
         1  38          0.0
         2  26          0.0
```

14 Check-in

14.0.1 You're all now experts:

- What `pipelines` are
- How to build a basic `pipeline`
- How each piece inside a `pipeline` works

14.0.2 Next up:

- Data Science Modeling Lifecycle
  - Model Development
  - Model Deployment

15 Data Science Modeling Lifecycle

16 Model Development

- Feature Engineering
- Model Selection
- Hyperparameter Optimization
- Cross Validation
- Publish

16.0.1 Can all of this be accomplished without `Pipelines`? Of course it can...

17 Why I heart `Pipelines` (Part I)

(emojis are hard in markdown)

Feature Engineering - Reuse `Transformers`! I use `CategoricalDummifier` in many projects - Readable transformation logic - Apply identical transformations to training and test (remember that mistake I made earlier) - You can put ANYTHING in a `Transformer` wrapper

```
In [26]: class xgb_ModelTransformer(TransformerMixin, BaseEstimator):
             def __init__(self, name, model, eval_metric=None):
                 self.name = name
                 self.model = model
                 self.eval_metric = eval_metric

             def fit(self, *args, **kwargs):
                 self.model.fit(eval_metric=self.eval_metric, *args, **kwargs)
                 return self

             def transform(self, X, **transform_params):
                 return pd.DataFrame(self.model.predict(X), columns=[self.name + '_
```

Model Selection - Modularity of `pipelines` makes changes clean and easy - Tracking the interactions and the hierarch of ensemblage models (models feeding into other models) - Not tracking intermediate data allows easy model stacking (PCA -> Random Forest)

Hyperparameter Optimization - `Pipelines` support Gridsearch!

17.0.1 Gridsearch!

Earlier pipeline we built

```
In [27]: pipeline.named_steps.keys()

Out[27]: ['factorextractor', 'fillna', 'converttodataframe', 'randomforestclassifie

In [28]: params = dict(randomforestclassifier__n_estimators= [100,250,300])
         grid_search = GridSearchCV(estimator=pipeline, param_grid=params, verbose=

In [29]: grid_search.fit(train_df, train_df[response_var].values)

Fitting 3 folds for each of 3 candidates, totalling 9 fits


[Parallel(n_jobs=1)]: Done   9 out of   9 | elapsed:    5.2s finished


Out[29]: GridSearchCV(cv=None, error_score='raise',
                estimator=Pipeline(steps=[('factorextractor', FactorExtractor(facto
                     max_depth=None, max_features='auto', ma...n_jobs=1,
                     oob_score=False, random_state=2065, verbose=0,
```

```
                warm_start=False))]),
           fit_params={}, iid=True, n_jobs=1,
           param_grid={'randomforestclassifier__n_estimators': [100, 250, 300]
           pre_dispatch='2*n_jobs', refit=True, scoring=None, verbose=1)
```

18 Publishing Models

Write our FIT model to disk

Publish to S3, github, a floppy disk, whatever! It's an object that we can do anything with!

```
In [30]: joblib.dump(pipeline, './our_fit_pipeline.pkl')

Out[30]: ['./our_fit_pipeline.pkl',
          './our_fit_pipeline.pkl_01.npy',
          './our_fit_pipeline.pkl_02.npy',
          './our_fit_pipeline.pkl_03.npy',
          './our_fit_pipeline.pkl_04.npy',
          './our_fit_pipeline.pkl_05.npy',
          './our_fit_pipeline.pkl_06.npy',
          './our_fit_pipeline.pkl_07.npy',
          './our_fit_pipeline.pkl_08.npy',
          './our_fit_pipeline.pkl_09.npy',
          './our_fit_pipeline.pkl_10.npy',
          './our_fit_pipeline.pkl_11.npy',
          './our_fit_pipeline.pkl_12.npy',
          './our_fit_pipeline.pkl_13.npy',
          './our_fit_pipeline.pkl_14.npy',
          './our_fit_pipeline.pkl_15.npy',
          './our_fit_pipeline.pkl_16.npy',
          './our_fit_pipeline.pkl_17.npy',
          './our_fit_pipeline.pkl_18.npy',
          './our_fit_pipeline.pkl_19.npy',
          './our_fit_pipeline.pkl_20.npy',
          './our_fit_pipeline.pkl_21.npy',
          './our_fit_pipeline.pkl_22.npy',
          './our_fit_pipeline.pkl_23.npy',
          './our_fit_pipeline.pkl_24.npy',
          './our_fit_pipeline.pkl_25.npy',
          './our_fit_pipeline.pkl_26.npy',
          './our_fit_pipeline.pkl_27.npy',
          './our_fit_pipeline.pkl_28.npy',
          './our_fit_pipeline.pkl_29.npy',
          './our_fit_pipeline.pkl_30.npy',
          './our_fit_pipeline.pkl_31.npy',
          './our_fit_pipeline.pkl_32.npy',
          './our_fit_pipeline.pkl_33.npy',
          './our_fit_pipeline.pkl_34.npy',
          './our_fit_pipeline.pkl_35.npy',
```

```
'./our_fit_pipeline.pkl_36.npy',
'./our_fit_pipeline.pkl_37.npy',
'./our_fit_pipeline.pkl_38.npy',
'./our_fit_pipeline.pkl_39.npy',
'./our_fit_pipeline.pkl_40.npy',
'./our_fit_pipeline.pkl_41.npy',
'./our_fit_pipeline.pkl_42.npy',
'./our_fit_pipeline.pkl_43.npy',
'./our_fit_pipeline.pkl_44.npy',
'./our_fit_pipeline.pkl_45.npy',
'./our_fit_pipeline.pkl_46.npy',
'./our_fit_pipeline.pkl_47.npy',
'./our_fit_pipeline.pkl_48.npy',
'./our_fit_pipeline.pkl_49.npy',
'./our_fit_pipeline.pkl_50.npy',
'./our_fit_pipeline.pkl_51.npy',
'./our_fit_pipeline.pkl_52.npy',
'./our_fit_pipeline.pkl_53.npy',
'./our_fit_pipeline.pkl_54.npy',
'./our_fit_pipeline.pkl_55.npy',
'./our_fit_pipeline.pkl_56.npy',
'./our_fit_pipeline.pkl_57.npy',
'./our_fit_pipeline.pkl_58.npy',
'./our_fit_pipeline.pkl_59.npy',
'./our_fit_pipeline.pkl_60.npy',
'./our_fit_pipeline.pkl_61.npy',
'./our_fit_pipeline.pkl_62.npy',
'./our_fit_pipeline.pkl_63.npy',
'./our_fit_pipeline.pkl_64.npy',
'./our_fit_pipeline.pkl_65.npy',
'./our_fit_pipeline.pkl_66.npy',
'./our_fit_pipeline.pkl_67.npy',
'./our_fit_pipeline.pkl_68.npy',
'./our_fit_pipeline.pkl_69.npy',
'./our_fit_pipeline.pkl_70.npy',
'./our_fit_pipeline.pkl_71.npy',
'./our_fit_pipeline.pkl_72.npy',
'./our_fit_pipeline.pkl_73.npy',
'./our_fit_pipeline.pkl_74.npy',
'./our_fit_pipeline.pkl_75.npy',
'./our_fit_pipeline.pkl_76.npy',
'./our_fit_pipeline.pkl_77.npy',
'./our_fit_pipeline.pkl_78.npy',
'./our_fit_pipeline.pkl_79.npy',
'./our_fit_pipeline.pkl_80.npy',
'./our_fit_pipeline.pkl_81.npy',
'./our_fit_pipeline.pkl_82.npy',
'./our_fit_pipeline.pkl_83.npy',
```

```
'./our_fit_pipeline.pkl_84.npy',
'./our_fit_pipeline.pkl_85.npy',
'./our_fit_pipeline.pkl_86.npy',
'./our_fit_pipeline.pkl_87.npy',
'./our_fit_pipeline.pkl_88.npy',
'./our_fit_pipeline.pkl_89.npy',
'./our_fit_pipeline.pkl_90.npy',
'./our_fit_pipeline.pkl_91.npy',
'./our_fit_pipeline.pkl_92.npy',
'./our_fit_pipeline.pkl_93.npy',
'./our_fit_pipeline.pkl_94.npy',
'./our_fit_pipeline.pkl_95.npy',
'./our_fit_pipeline.pkl_96.npy',
'./our_fit_pipeline.pkl_97.npy',
'./our_fit_pipeline.pkl_98.npy',
'./our_fit_pipeline.pkl_99.npy',
'./our_fit_pipeline.pkl_100.npy',
'./our_fit_pipeline.pkl_101.npy',
'./our_fit_pipeline.pkl_102.npy',
'./our_fit_pipeline.pkl_103.npy',
'./our_fit_pipeline.pkl_104.npy',
'./our_fit_pipeline.pkl_105.npy',
'./our_fit_pipeline.pkl_106.npy',
'./our_fit_pipeline.pkl_107.npy',
'./our_fit_pipeline.pkl_108.npy',
'./our_fit_pipeline.pkl_109.npy',
'./our_fit_pipeline.pkl_110.npy',
'./our_fit_pipeline.pkl_111.npy',
'./our_fit_pipeline.pkl_112.npy',
'./our_fit_pipeline.pkl_113.npy',
'./our_fit_pipeline.pkl_114.npy',
'./our_fit_pipeline.pkl_115.npy',
'./our_fit_pipeline.pkl_116.npy',
'./our_fit_pipeline.pkl_117.npy',
'./our_fit_pipeline.pkl_118.npy',
'./our_fit_pipeline.pkl_119.npy',
'./our_fit_pipeline.pkl_120.npy',
'./our_fit_pipeline.pkl_121.npy',
'./our_fit_pipeline.pkl_122.npy',
'./our_fit_pipeline.pkl_123.npy',
'./our_fit_pipeline.pkl_124.npy',
'./our_fit_pipeline.pkl_125.npy',
'./our_fit_pipeline.pkl_126.npy',
'./our_fit_pipeline.pkl_127.npy',
'./our_fit_pipeline.pkl_128.npy',
'./our_fit_pipeline.pkl_129.npy',
'./our_fit_pipeline.pkl_130.npy',
'./our_fit_pipeline.pkl_131.npy',
```

```
'./our_fit_pipeline.pkl_132.npy',
'./our_fit_pipeline.pkl_133.npy',
'./our_fit_pipeline.pkl_134.npy',
'./our_fit_pipeline.pkl_135.npy',
'./our_fit_pipeline.pkl_136.npy',
'./our_fit_pipeline.pkl_137.npy',
'./our_fit_pipeline.pkl_138.npy',
'./our_fit_pipeline.pkl_139.npy',
'./our_fit_pipeline.pkl_140.npy',
'./our_fit_pipeline.pkl_141.npy',
'./our_fit_pipeline.pkl_142.npy',
'./our_fit_pipeline.pkl_143.npy',
'./our_fit_pipeline.pkl_144.npy',
'./our_fit_pipeline.pkl_145.npy',
'./our_fit_pipeline.pkl_146.npy',
'./our_fit_pipeline.pkl_147.npy',
'./our_fit_pipeline.pkl_148.npy',
'./our_fit_pipeline.pkl_149.npy',
'./our_fit_pipeline.pkl_150.npy',
'./our_fit_pipeline.pkl_151.npy',
'./our_fit_pipeline.pkl_152.npy',
'./our_fit_pipeline.pkl_153.npy',
'./our_fit_pipeline.pkl_154.npy',
'./our_fit_pipeline.pkl_155.npy',
'./our_fit_pipeline.pkl_156.npy',
'./our_fit_pipeline.pkl_157.npy',
'./our_fit_pipeline.pkl_158.npy',
'./our_fit_pipeline.pkl_159.npy',
'./our_fit_pipeline.pkl_160.npy',
'./our_fit_pipeline.pkl_161.npy',
'./our_fit_pipeline.pkl_162.npy',
'./our_fit_pipeline.pkl_163.npy',
'./our_fit_pipeline.pkl_164.npy',
'./our_fit_pipeline.pkl_165.npy',
'./our_fit_pipeline.pkl_166.npy',
'./our_fit_pipeline.pkl_167.npy',
'./our_fit_pipeline.pkl_168.npy',
'./our_fit_pipeline.pkl_169.npy',
'./our_fit_pipeline.pkl_170.npy',
'./our_fit_pipeline.pkl_171.npy',
'./our_fit_pipeline.pkl_172.npy',
'./our_fit_pipeline.pkl_173.npy',
'./our_fit_pipeline.pkl_174.npy',
'./our_fit_pipeline.pkl_175.npy',
'./our_fit_pipeline.pkl_176.npy',
'./our_fit_pipeline.pkl_177.npy',
'./our_fit_pipeline.pkl_178.npy',
'./our_fit_pipeline.pkl_179.npy',
```

```
'./our_fit_pipeline.pkl_180.npy',
'./our_fit_pipeline.pkl_181.npy',
'./our_fit_pipeline.pkl_182.npy',
'./our_fit_pipeline.pkl_183.npy',
'./our_fit_pipeline.pkl_184.npy',
'./our_fit_pipeline.pkl_185.npy',
'./our_fit_pipeline.pkl_186.npy',
'./our_fit_pipeline.pkl_187.npy',
'./our_fit_pipeline.pkl_188.npy',
'./our_fit_pipeline.pkl_189.npy',
'./our_fit_pipeline.pkl_190.npy',
'./our_fit_pipeline.pkl_191.npy',
'./our_fit_pipeline.pkl_192.npy',
'./our_fit_pipeline.pkl_193.npy',
'./our_fit_pipeline.pkl_194.npy',
'./our_fit_pipeline.pkl_195.npy',
'./our_fit_pipeline.pkl_196.npy',
'./our_fit_pipeline.pkl_197.npy',
'./our_fit_pipeline.pkl_198.npy',
'./our_fit_pipeline.pkl_199.npy',
'./our_fit_pipeline.pkl_200.npy',
'./our_fit_pipeline.pkl_201.npy',
'./our_fit_pipeline.pkl_202.npy',
'./our_fit_pipeline.pkl_203.npy',
'./our_fit_pipeline.pkl_204.npy',
'./our_fit_pipeline.pkl_205.npy',
'./our_fit_pipeline.pkl_206.npy',
'./our_fit_pipeline.pkl_207.npy',
'./our_fit_pipeline.pkl_208.npy',
'./our_fit_pipeline.pkl_209.npy',
'./our_fit_pipeline.pkl_210.npy',
'./our_fit_pipeline.pkl_211.npy',
'./our_fit_pipeline.pkl_212.npy',
'./our_fit_pipeline.pkl_213.npy',
'./our_fit_pipeline.pkl_214.npy',
'./our_fit_pipeline.pkl_215.npy',
'./our_fit_pipeline.pkl_216.npy',
'./our_fit_pipeline.pkl_217.npy',
'./our_fit_pipeline.pkl_218.npy',
'./our_fit_pipeline.pkl_219.npy',
'./our_fit_pipeline.pkl_220.npy',
'./our_fit_pipeline.pkl_221.npy',
'./our_fit_pipeline.pkl_222.npy',
'./our_fit_pipeline.pkl_223.npy',
'./our_fit_pipeline.pkl_224.npy',
'./our_fit_pipeline.pkl_225.npy',
'./our_fit_pipeline.pkl_226.npy',
'./our_fit_pipeline.pkl_227.npy',
```

```
'./our_fit_pipeline.pkl_228.npy',
'./our_fit_pipeline.pkl_229.npy',
'./our_fit_pipeline.pkl_230.npy',
'./our_fit_pipeline.pkl_231.npy',
'./our_fit_pipeline.pkl_232.npy',
'./our_fit_pipeline.pkl_233.npy',
'./our_fit_pipeline.pkl_234.npy',
'./our_fit_pipeline.pkl_235.npy',
'./our_fit_pipeline.pkl_236.npy',
'./our_fit_pipeline.pkl_237.npy',
'./our_fit_pipeline.pkl_238.npy',
'./our_fit_pipeline.pkl_239.npy',
'./our_fit_pipeline.pkl_240.npy',
'./our_fit_pipeline.pkl_241.npy',
'./our_fit_pipeline.pkl_242.npy',
'./our_fit_pipeline.pkl_243.npy',
'./our_fit_pipeline.pkl_244.npy',
'./our_fit_pipeline.pkl_245.npy',
'./our_fit_pipeline.pkl_246.npy',
'./our_fit_pipeline.pkl_247.npy',
'./our_fit_pipeline.pkl_248.npy',
'./our_fit_pipeline.pkl_249.npy',
'./our_fit_pipeline.pkl_250.npy',
'./our_fit_pipeline.pkl_251.npy',
'./our_fit_pipeline.pkl_252.npy',
'./our_fit_pipeline.pkl_253.npy',
'./our_fit_pipeline.pkl_254.npy',
'./our_fit_pipeline.pkl_255.npy',
'./our_fit_pipeline.pkl_256.npy',
'./our_fit_pipeline.pkl_257.npy',
'./our_fit_pipeline.pkl_258.npy',
'./our_fit_pipeline.pkl_259.npy',
'./our_fit_pipeline.pkl_260.npy',
'./our_fit_pipeline.pkl_261.npy',
'./our_fit_pipeline.pkl_262.npy',
'./our_fit_pipeline.pkl_263.npy',
'./our_fit_pipeline.pkl_264.npy',
'./our_fit_pipeline.pkl_265.npy',
'./our_fit_pipeline.pkl_266.npy',
'./our_fit_pipeline.pkl_267.npy',
'./our_fit_pipeline.pkl_268.npy',
'./our_fit_pipeline.pkl_269.npy',
'./our_fit_pipeline.pkl_270.npy',
'./our_fit_pipeline.pkl_271.npy',
'./our_fit_pipeline.pkl_272.npy',
'./our_fit_pipeline.pkl_273.npy',
'./our_fit_pipeline.pkl_274.npy',
'./our_fit_pipeline.pkl_275.npy',
```

```
'./our_fit_pipeline.pkl_276.npy',
'./our_fit_pipeline.pkl_277.npy',
'./our_fit_pipeline.pkl_278.npy',
'./our_fit_pipeline.pkl_279.npy',
'./our_fit_pipeline.pkl_280.npy',
'./our_fit_pipeline.pkl_281.npy',
'./our_fit_pipeline.pkl_282.npy',
'./our_fit_pipeline.pkl_283.npy',
'./our_fit_pipeline.pkl_284.npy',
'./our_fit_pipeline.pkl_285.npy',
'./our_fit_pipeline.pkl_286.npy',
'./our_fit_pipeline.pkl_287.npy',
'./our_fit_pipeline.pkl_288.npy',
'./our_fit_pipeline.pkl_289.npy',
'./our_fit_pipeline.pkl_290.npy',
'./our_fit_pipeline.pkl_291.npy',
'./our_fit_pipeline.pkl_292.npy',
'./our_fit_pipeline.pkl_293.npy',
'./our_fit_pipeline.pkl_294.npy',
'./our_fit_pipeline.pkl_295.npy',
'./our_fit_pipeline.pkl_296.npy',
'./our_fit_pipeline.pkl_297.npy',
'./our_fit_pipeline.pkl_298.npy',
'./our_fit_pipeline.pkl_299.npy',
'./our_fit_pipeline.pkl_300.npy',
'./our_fit_pipeline.pkl_301.npy',
'./our_fit_pipeline.pkl_302.npy',
'./our_fit_pipeline.pkl_303.npy',
'./our_fit_pipeline.pkl_304.npy',
'./our_fit_pipeline.pkl_305.npy',
'./our_fit_pipeline.pkl_306.npy',
'./our_fit_pipeline.pkl_307.npy',
'./our_fit_pipeline.pkl_308.npy',
'./our_fit_pipeline.pkl_309.npy',
'./our_fit_pipeline.pkl_310.npy',
'./our_fit_pipeline.pkl_311.npy',
'./our_fit_pipeline.pkl_312.npy',
'./our_fit_pipeline.pkl_313.npy',
'./our_fit_pipeline.pkl_314.npy',
'./our_fit_pipeline.pkl_315.npy',
'./our_fit_pipeline.pkl_316.npy',
'./our_fit_pipeline.pkl_317.npy',
'./our_fit_pipeline.pkl_318.npy',
'./our_fit_pipeline.pkl_319.npy',
'./our_fit_pipeline.pkl_320.npy',
'./our_fit_pipeline.pkl_321.npy',
'./our_fit_pipeline.pkl_322.npy',
'./our_fit_pipeline.pkl_323.npy',
```

```
'./our_fit_pipeline.pkl_324.npy',
'./our_fit_pipeline.pkl_325.npy',
'./our_fit_pipeline.pkl_326.npy',
'./our_fit_pipeline.pkl_327.npy',
'./our_fit_pipeline.pkl_328.npy',
'./our_fit_pipeline.pkl_329.npy',
'./our_fit_pipeline.pkl_330.npy',
'./our_fit_pipeline.pkl_331.npy',
'./our_fit_pipeline.pkl_332.npy',
'./our_fit_pipeline.pkl_333.npy',
'./our_fit_pipeline.pkl_334.npy',
'./our_fit_pipeline.pkl_335.npy',
'./our_fit_pipeline.pkl_336.npy',
'./our_fit_pipeline.pkl_337.npy',
'./our_fit_pipeline.pkl_338.npy',
'./our_fit_pipeline.pkl_339.npy',
'./our_fit_pipeline.pkl_340.npy',
'./our_fit_pipeline.pkl_341.npy',
'./our_fit_pipeline.pkl_342.npy',
'./our_fit_pipeline.pkl_343.npy',
'./our_fit_pipeline.pkl_344.npy',
'./our_fit_pipeline.pkl_345.npy',
'./our_fit_pipeline.pkl_346.npy',
'./our_fit_pipeline.pkl_347.npy',
'./our_fit_pipeline.pkl_348.npy',
'./our_fit_pipeline.pkl_349.npy',
'./our_fit_pipeline.pkl_350.npy',
'./our_fit_pipeline.pkl_351.npy',
'./our_fit_pipeline.pkl_352.npy',
'./our_fit_pipeline.pkl_353.npy',
'./our_fit_pipeline.pkl_354.npy',
'./our_fit_pipeline.pkl_355.npy',
'./our_fit_pipeline.pkl_356.npy',
'./our_fit_pipeline.pkl_357.npy',
'./our_fit_pipeline.pkl_358.npy',
'./our_fit_pipeline.pkl_359.npy',
'./our_fit_pipeline.pkl_360.npy',
'./our_fit_pipeline.pkl_361.npy',
'./our_fit_pipeline.pkl_362.npy',
'./our_fit_pipeline.pkl_363.npy',
'./our_fit_pipeline.pkl_364.npy',
'./our_fit_pipeline.pkl_365.npy',
'./our_fit_pipeline.pkl_366.npy',
'./our_fit_pipeline.pkl_367.npy',
'./our_fit_pipeline.pkl_368.npy',
'./our_fit_pipeline.pkl_369.npy',
'./our_fit_pipeline.pkl_370.npy',
'./our_fit_pipeline.pkl_371.npy',
```

```
'./our_fit_pipeline.pkl_372.npy',
'./our_fit_pipeline.pkl_373.npy',
'./our_fit_pipeline.pkl_374.npy',
'./our_fit_pipeline.pkl_375.npy',
'./our_fit_pipeline.pkl_376.npy',
'./our_fit_pipeline.pkl_377.npy',
'./our_fit_pipeline.pkl_378.npy',
'./our_fit_pipeline.pkl_379.npy',
'./our_fit_pipeline.pkl_380.npy',
'./our_fit_pipeline.pkl_381.npy',
'./our_fit_pipeline.pkl_382.npy',
'./our_fit_pipeline.pkl_383.npy',
'./our_fit_pipeline.pkl_384.npy',
'./our_fit_pipeline.pkl_385.npy',
'./our_fit_pipeline.pkl_386.npy',
'./our_fit_pipeline.pkl_387.npy',
'./our_fit_pipeline.pkl_388.npy',
'./our_fit_pipeline.pkl_389.npy',
'./our_fit_pipeline.pkl_390.npy',
'./our_fit_pipeline.pkl_391.npy',
'./our_fit_pipeline.pkl_392.npy',
'./our_fit_pipeline.pkl_393.npy',
'./our_fit_pipeline.pkl_394.npy',
'./our_fit_pipeline.pkl_395.npy',
'./our_fit_pipeline.pkl_396.npy',
'./our_fit_pipeline.pkl_397.npy',
'./our_fit_pipeline.pkl_398.npy',
'./our_fit_pipeline.pkl_399.npy',
'./our_fit_pipeline.pkl_400.npy',
'./our_fit_pipeline.pkl_401.npy']
```

19 Deployment

19.0.1 The 3-ish slides you've been waiting 30 minutes for

19.0.2 The Handoff!

Now that you've got an entire `Pipeline` saved to disk you can pass it anywhere

All your deployed model needs to do is be loaded into memory and have the `predict` method called.

19.0.3 YOU DO NOT HAVE TO RETRAIN YOUR MODEL EVERY TIME YOU WANT TO SCORE NEW DATA

19.0.4 (Kevin did you mention this is the most common mistake you see? Because it's super important for you to emphasize if you forgot to)

19.0.5 Loading from Disk

```
In [31]: some_totally_random_model = joblib.load('./our_fit_pipeline.pkl')
         some_totally_random_model.named_steps.keys()

Out[31]: ['factorextractor', 'fillna', 'converttodataframe', 'randomforestclassifie
```

We can even see what parameters went into each step!

```
In [32]: some_totally_random_model.steps[1]

Out[32]: ('fillna', FillNA(NA_replacement=28.0))
```

Maybe not a big deal if it's been minutes since we set the replacement to 28.

HUGE deal if it's been 3 months, the coworker who wrote it has quit and the development code is in a jumble.

Just imagine the headache of wading through 20+ transformations done in non-`Pipeline` code.

20 Data Engineer != Data Scientist

Passing a standard object will make you many friends - Engineers don't want to build a bespoke process for every model developed - A standard deployment approach is easy if every object has the same methods - Why should an engineer build a seperate process to predict fraud and customer clickthrough rates? - Data Engineers are specialists too - If someone's on vacation their coworkers can fill in with little training

Process - Load model - Load input data - model.predict(data) - write ouput to db

21 Why I heart `Pipelines` (Part II)

- Tests can be written
- Models are backed up in version control
  - Rerunning the model version 2.6 that you ran last Dec is a snap
  - Peers can check out and test your model
- Open Source
  - You can write your own `DataFrameUnion` extention
  - You can hire folks already familiar with `Pipelines`
  - You can collaborate across teams, departments, even companies

22 Things I hope you walk away with

- What Sklearn `Pipelines` kinda look like
- Where you could benefit from using `Pipelines`
- That I think `Pipelines` are awesome!
- That my Father still has no idea what I do

In [ ]: