

Data Science & ML Course

Lesson #23 Kaggle Fundamentals

Ivanovitch Silva
December, 2018



Update from repository

```
git clone https://github.com/ivanovitchm/datascience2machinelearning.git
```

Or

```
git pull
```



Agenda

1. Getting Started with Kaggle
2. Feature Preparation, Selection and Engineering
3. Model Selection and Tuning
4. Creating a Kaggle Workflow

Introduction to Kaggle

kaggle

- Approach a Kaggle competition.
- Explore the competition data and learn about the competition topic.
- Prepare data for machine learning.
- Train a model.
- Measure the accuracy of your model.
- Prepare and make your first Kaggle submission.





Titanic: Machine Learning from Disaster


Start here! Predict survival on the Titanic and get familiar with ML basics



Kaggle · 10,331 teams · Ongoing

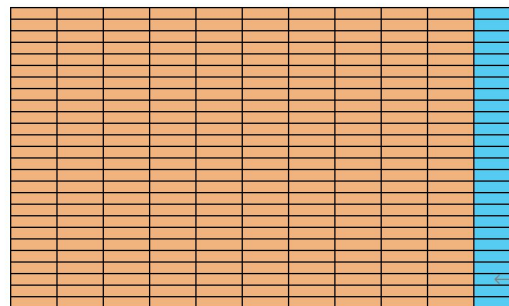
[Overview](#) [Data](#) [Kernels](#) [Discussion](#) [Leaderboard](#) [Rules](#)

Data Sources

 gender_submission.csv 418 x 2

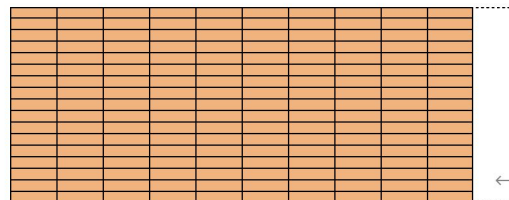
 test.csv 418 x 11

 train.csv 891 x 12



Training Set

Survival Data for
each passenger



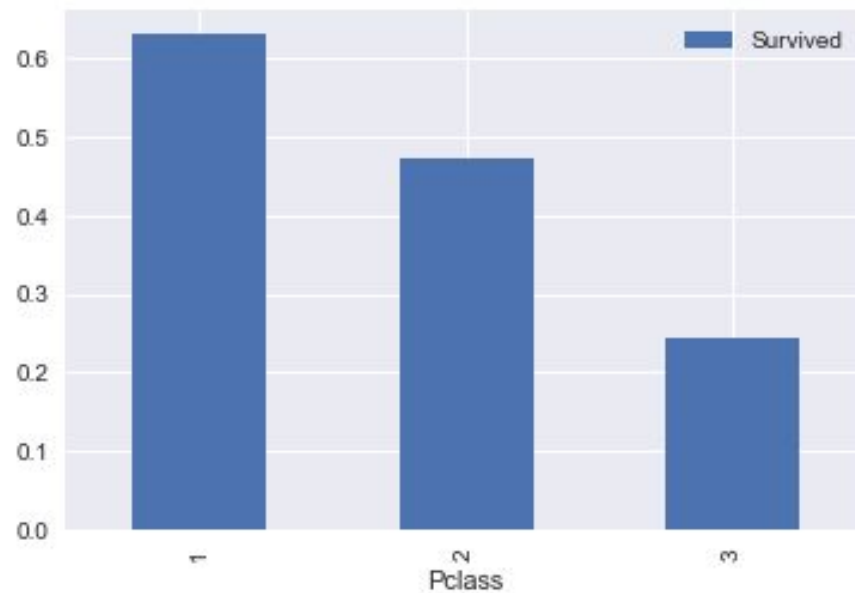
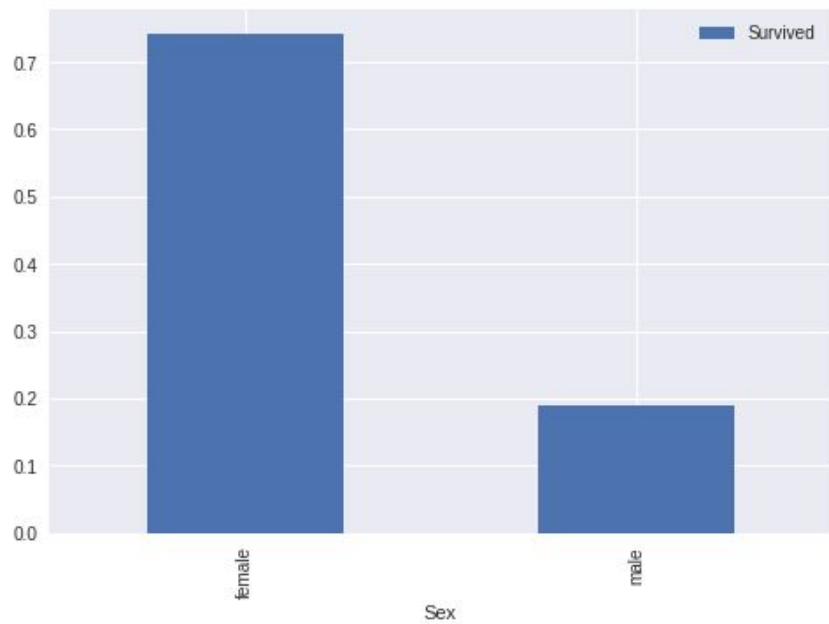
Testing Set

No Survival Data

Exploring the Data

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Exploring the Data



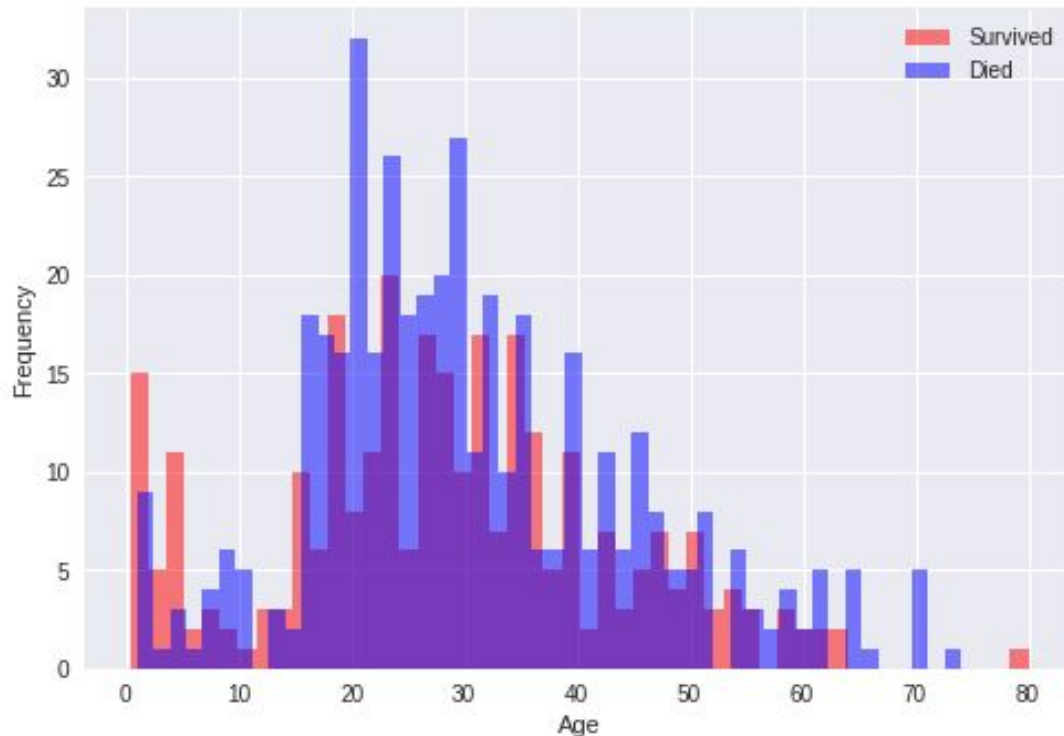
Exploring and Converting the Age Column

```
1 train["Age"].describe().
```

```
count    714.000000
mean     29.699118
std      14.526497
min       0.420000
25%      20.125000
50%      28.000000
75%      38.000000
max      80.000000
Name: Age, dtype: float64
```

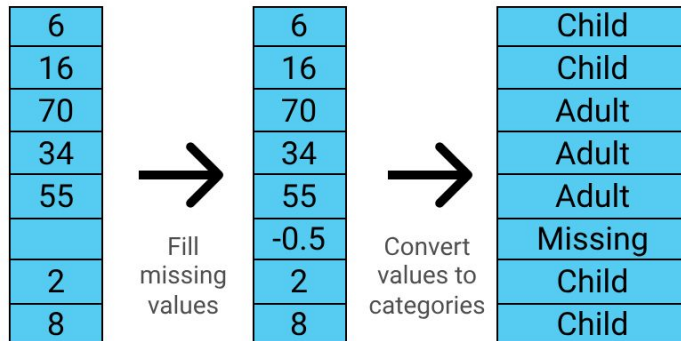
```
1 train.Age.isnull().sum().
```

177

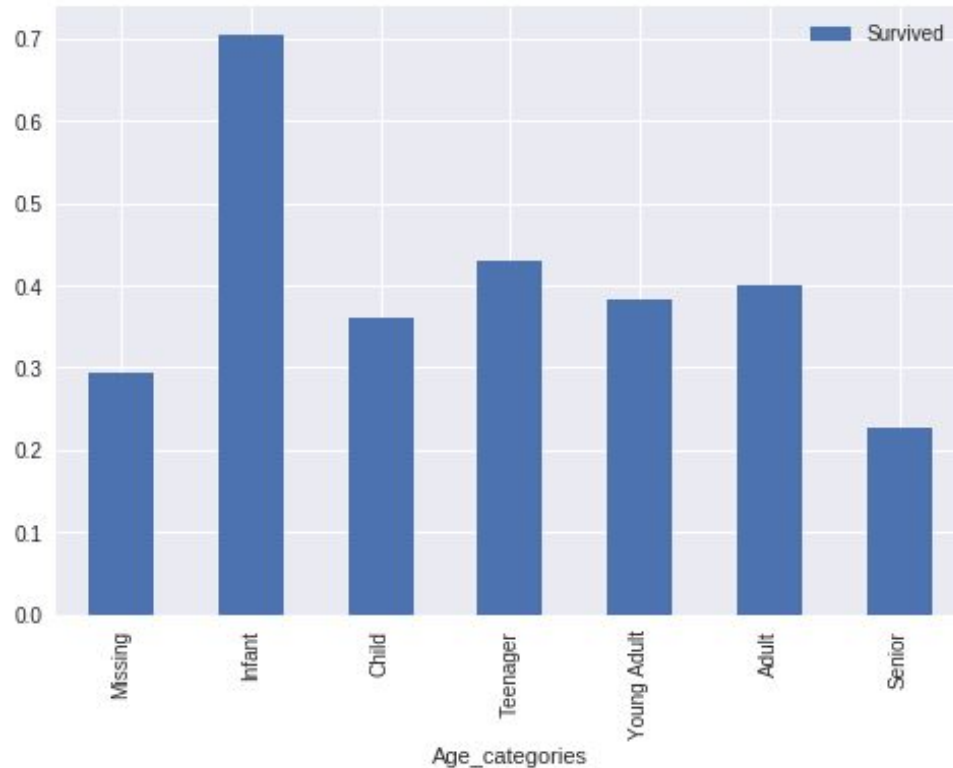


Exploring and Converting the Age Column

```
def process_age(df, cut_points, label_names):  
    df["Age"] = df["Age"].fillna(-0.5)  
    df["Age_categories"] = pd.cut(df["Age"],  
                                  cut_points,  
                                  labels=label_names)  
  
    return df  
  
cut_points = [-1, 0, 18, 100]  
label_names = ["Missing", "Child", "Adult"]  
train = process_age(train, cut_points, label_names)  
test = process_age(test, cut_points, label_names)
```



Exploring and Converting the Age Column



Preparing our Data for Machine Learning

- Sex
- Pclass
- Age
- Age_categories
- Before we build our model, we need to prepare these columns for machine learning.
- Most machine learning algorithms can't understand text labels, so we have to convert our values into numbers.

Preparing our Data for Machine Learning

Pclass	Pclass_1	Pclass_2	Pclass_3
3	0	0	1
1	1	0	0
3	0	0	1
1	1	0	0
3	0	0	1
3	0	0	1
1	1	0	0
3	0	0	1
3	0	0	1
2	0	1	0

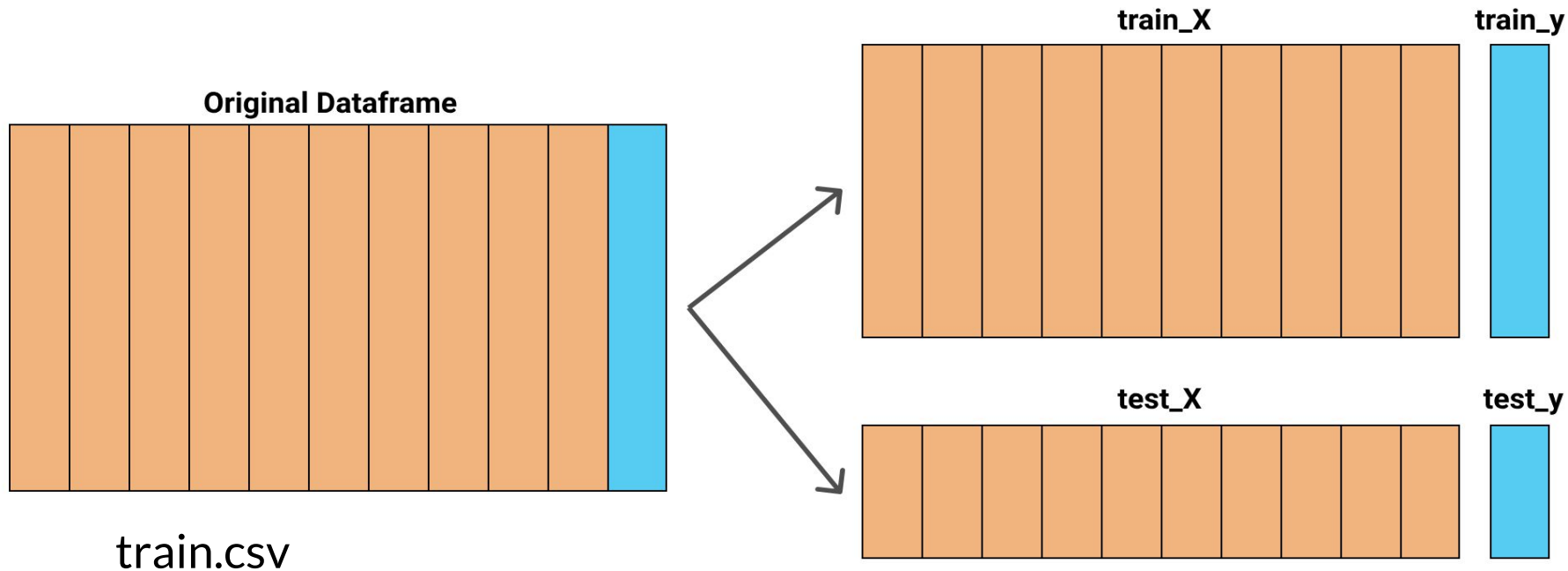
```
def create_dummies(df,column_name):  
    dummies = pd.get_dummies(df[column_name],  
                              prefix=column_name)  
    df = pd.concat([df,dummies],axis=1)  
    return df  
train = create_dummies(train,"Pclass")  
test = create_dummies(test,"Pclass")
```

Creating our First Machine Learning Model

The scikit-learn workflow consists of **four main steps**:

- **Instantiate** (or create) the specific machine learning model you want to use
- **Fit** the model to the training data
- Use the model to make **predictions**
- **Evaluate** the accuracy of the predictions

Creating our First Machine Learning Model



Creating our First Machine Learning Model

```
1 holdout = test # from now on we will refer to this
2               # dataframe as the holdout data
3
4 from sklearn.model_selection import train_test_split
5
6 columns = ['Pclass_1', 'Pclass_2', 'Pclass_3', 'Sex_female', 'Sex_male',
7            'Age_categories_Missing', 'Age_categories_Infant',
8            'Age_categories_Child', 'Age_categories_Teenager',
9            'Age_categories_Young Adult', 'Age_categories_Adult',
10           'Age_categories_Senior']
11
12 all_X = train[columns]
13 all_y = train['Survived']
14
15 train_X, test_X, train_y, test_y = train_test_split(
16     all_X, all_y, test_size=0.20, random_state=0)
```

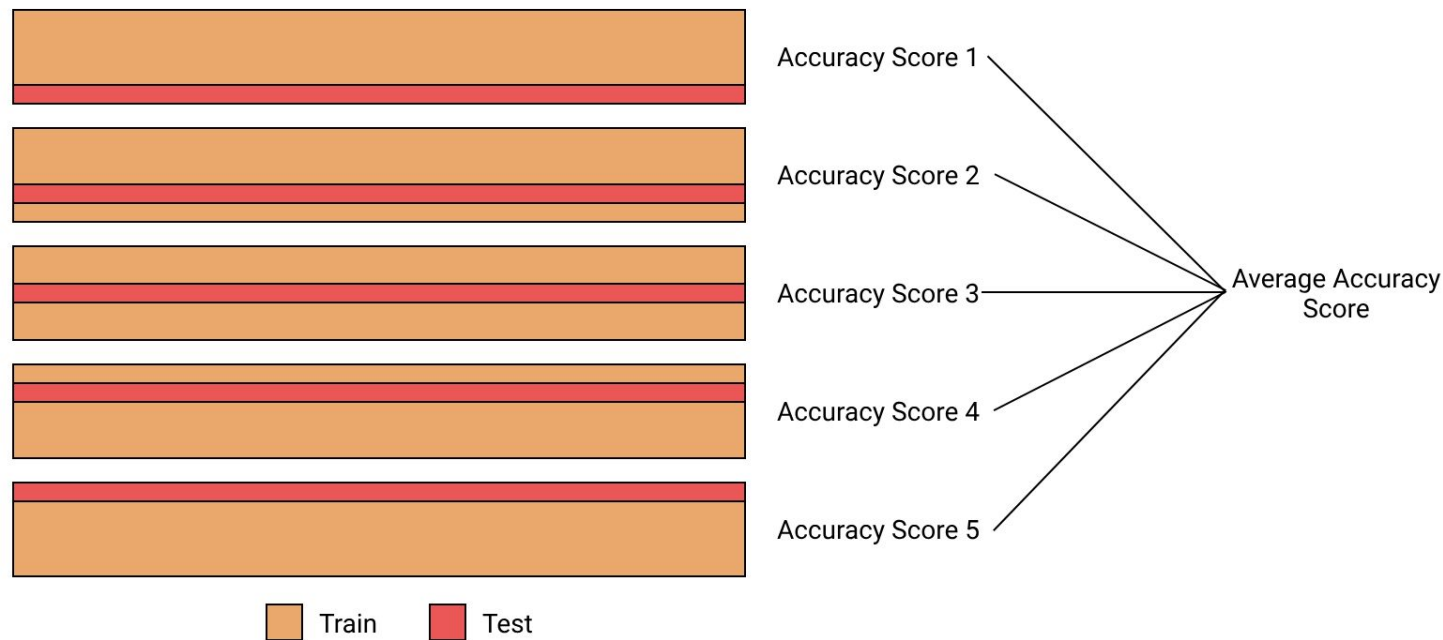
Making Predictions and Measuring their Accuracy

```
1 from sklearn.metrics import accuracy_score
2
3 lr = LogisticRegression()
4 lr.fit(train_X, train_y)
5 predictions = lr.predict(test_X)
6
7 accuracy = accuracy_score(test_y, predictions)
8 print(accuracy)
```

0.8100558659217877

Our model's prediction	The actual value	Correct
0	0	Yes
1	0	No
0	1	No
1	1	Yes
1	1	Yes

Using cross validation for more accurate error measurement



Using cross validation for more accurate error measurement

```
1 from sklearn.model_selection import cross_val_score
2 import numpy as np
3
4 lr = LogisticRegression()
5 scores = cross_val_score(lr, all_X, all_y, cv=10)
6 accuracy = np.mean(scores)
7 print(scores)
8 print(accuracy)
```

```
[0.8          0.81111111 0.7752809  0.87640449 0.80898876 0.78651685
 0.76404494 0.76404494 0.83146067 0.80681818]
0.8024670865963002
```

Making Predictions on Unseen Data













```
1 columns = ['Pclass_1', 'Pclass_2', 'Pclass_3', 'Sex_female', 'Sex_male',  
2           'Age_categories_Missing', 'Age_categories_Infant',  
3           'Age_categories_Child', 'Age_categories_Teenager',  
4           'Age_categories_Young Adult', 'Age_categories_Adult',  
5           'Age_categories_Senior']  
6  
7 lr = LogisticRegression()  
8  
9 lr.fit(all_X, all_y)  
10  
11 holdout_predictions = lr.predict(holdout[columns]).
```

Creating a Submission File

PassengerId	Survived
892	0
893	1
894	0

```
1 holdout_ids = holdout["PassengerId"]
2
3 submission_df = {"PassengerId": holdout_ids,
4                  "Survived": holdout_predictions}
5 submission = pd.DataFrame(submission_df)
6
7 submission.to_csv("submission.csv", index=False)
```

Making our first submission to kaggle

7999	new	kingsning		0.75598	2	13h
8000	new	hbruhn		0.75598	2	13h
8001	new	adam ardiansyah		0.75598	1	10h
8002	new	Penguin9		0.75598	10	6h
8003	new	Meghna Ayyar		0.75598	4	10h
8004	new	gauravgupta12		0.75598	3	3h
8005	new	Raph_Partouche		0.75598	1	3h
8006	new	IvanovitchSilva		0.75598	1	~10s
Your Best Entry ↑ Your submission scored 0.75598, which is not an improvement of your best score. Keep trying!						
8007	▼ 878	Sudarshan Kannan		0.75119	7	2mo
8008	▼ 878	Raunak Kumar		0.75119	1	2mo
8009	▼ 878	cyrilb38		0.75119	1	2mo
8010	▼ 878	simon 27		0.75119	3	2mo

Next Steps

- **Improving the features:**
 - Feature Engineering: Create new features from the existing data.
 - Feature Selection: Select the most relevant features to reduce noise and overfitting.
- **Improving the model:**
 - Model Selection: Try a variety of models to improve performance.
 - Hyperparameter Optimization: Optimize the settings within each particular machine learning model.

1. Getting Started with Kaggle.ipynb



Feature preparation: age, pclass, sex

```
def process_age(df):
    df["Age"] = df["Age"].fillna(-0.5)
    cut_points = [-1,0,5,12,18,35,60,100]
    label_names = ["Missing","Infant","Child",
                  "Teenager","Young Adult","Adult","Senior"]
    df["Age_categories"] = pd.cut(df["Age"],cut_points,labels=label_names)
    return df

def create_dummies(df,column_name):
    dummies = pd.get_dummies(df[column_name],prefix=column_name)
    df = pd.concat([df,dummies],axis=1)
    return df

train = process_age(train)
holdout = process_age(holdout)

for column in ["Age_categories","Pclass","Sex"]:
    train = create_dummies(train,column)
    holdout = create_dummies(holdout,column)
```


Preparing more features

	SibSp	Parch	Fare	Cabin	Embarked
count	891.000000	891.000000	891.000000	204	889
unique	NaN	NaN	NaN	147	3
top	NaN	NaN	NaN	C23 C25 C27	S
freq	NaN	NaN	NaN	4	644
mean	0.523008	0.381594	32.204208	NaN	NaN
std	1.102743	0.806057	49.693429	NaN	NaN
min	0.000000	0.000000	0.000000	NaN	NaN
25%	0.000000	0.000000	7.910400	NaN	NaN
50%	0.000000	0.000000	14.454200	NaN	NaN
75%	1.000000	0.000000	31.000000	NaN	NaN
max	8.000000	6.000000	512.329200	NaN	NaN

```
1 train[colums].isnull().sum()
```

```
SibSp      0
Parch      0
Fare       0
Cabin     687
Embarked   2
dtype: int64
```

```
1 holdout[colums].isnull().sum()
```

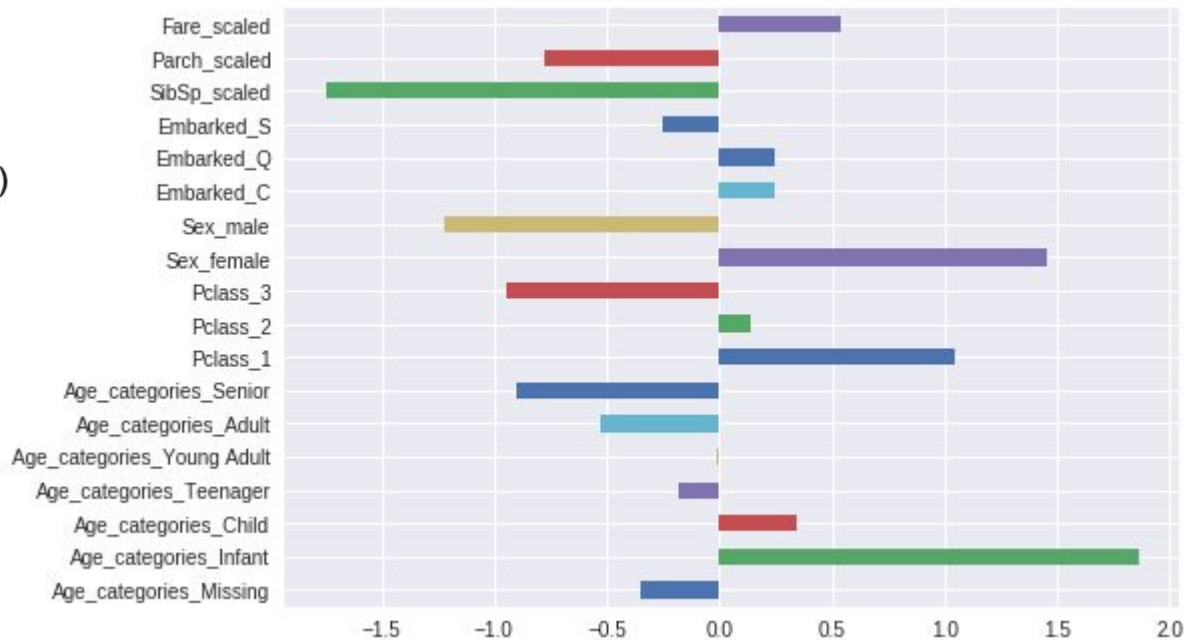
```
SibSp      0
Parch      0
Fare       1
Cabin     327
Embarked   0
dtype: int64
```

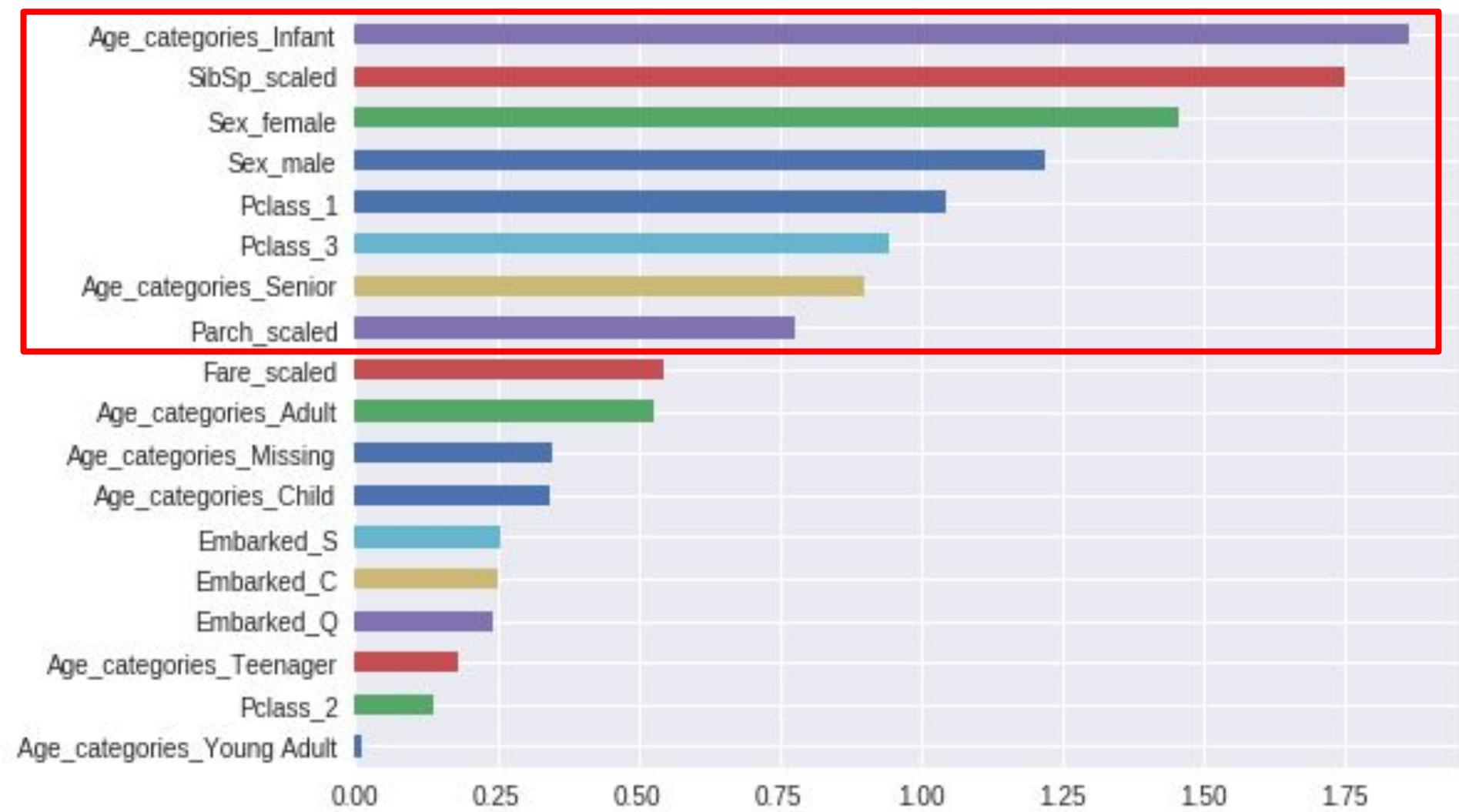
Preparing more features

```
1 from sklearn.preprocessing import minmax_scale
2 # The holdout set has a missing value in the Fare column which
3 # we'll fill with the mean.
4 holdout["Fare"] = holdout["Fare"].fillna(train["Fare"].mean())
5 columns = ["SibSp", "Parch", "Fare"]
6
7 train["Embarked"] = train["Embarked"].fillna("S")
8 holdout["Embarked"] = holdout["Embarked"].fillna("S")
9
10 train = create_dummies(train, "Embarked")
11 holdout = create_dummies(holdout, "Embarked")
12
13 for col in columns:
14     train[col + "_scaled"] = minmax_scale(train[col])
15     holdout[col + "_scaled"] = minmax_scale(holdout[col])
```

Determining the most relevant features

```
lr = LogisticRegression()  
lr.fit(train_X,train_y)  
coefficients = lr.coef_
```





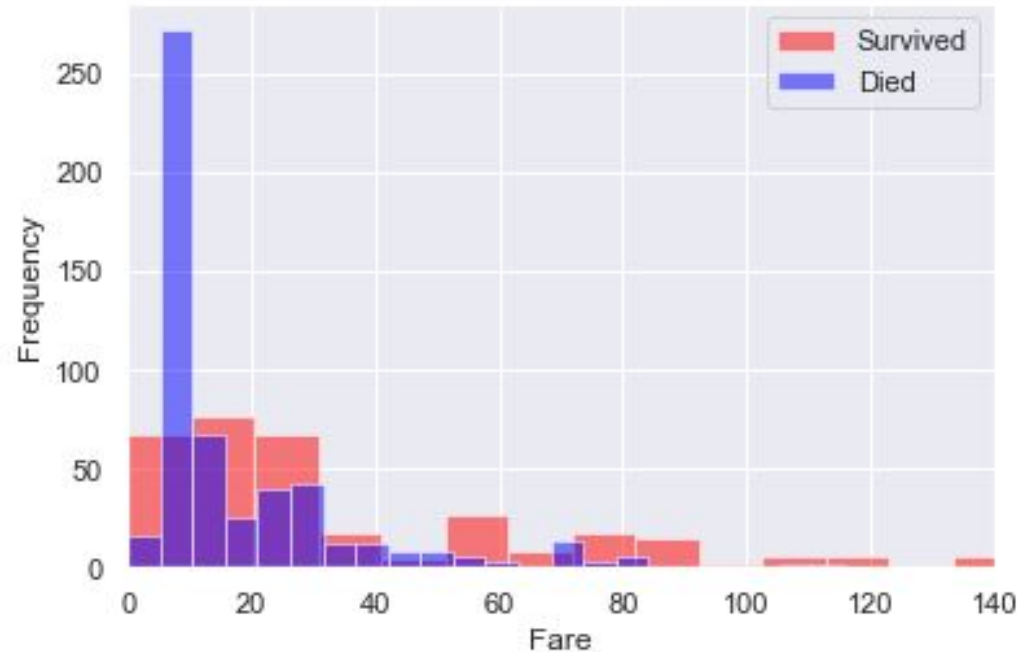
Training a model using relevant features

```
1 from sklearn.model_selection import cross_val_score
2
3 columns = ['Age_categories_Infant', 'SibSp_scaled', 'Sex_female', 'Sex_male',
4           'Pclass_1', 'Pclass_3', 'Age_categories_Senior', 'Parch_scaled']
5 all_X = train[columns]
6 all_y = train['Survived'].
7
8 lr = LogisticRegression()
9 scores = cross_val_score(lr, all_X, all_y, cv=10)
10 accuracy = scores.mean()
11 print(accuracy)
```

0.8148019521053229

When you submit it to Kaggle, you'll see
that the score is **77.33%**

Engineering a new feature using binning



```
def process_fare(df, cut_points, label_names):  
    df["Fare_categories"] = pd.cut(df["Fare"],  
                                   cut_points,  
                                   labels=label_names)  
  
    return df  
  
cut_points = [0, 12, 50, 100, 1000]  
label_names = ["0-12", "12-50", "50-100", "100+"]  
  
train = process_fare(train, cut_points, label_names)  
holdout = process_fare(holdout, cut_points, label_names)  
  
train = create_dummies(train, "Fare_categories")  
holdout = create_dummies(holdout, "Fare_categories")
```

Engineering features from text columns

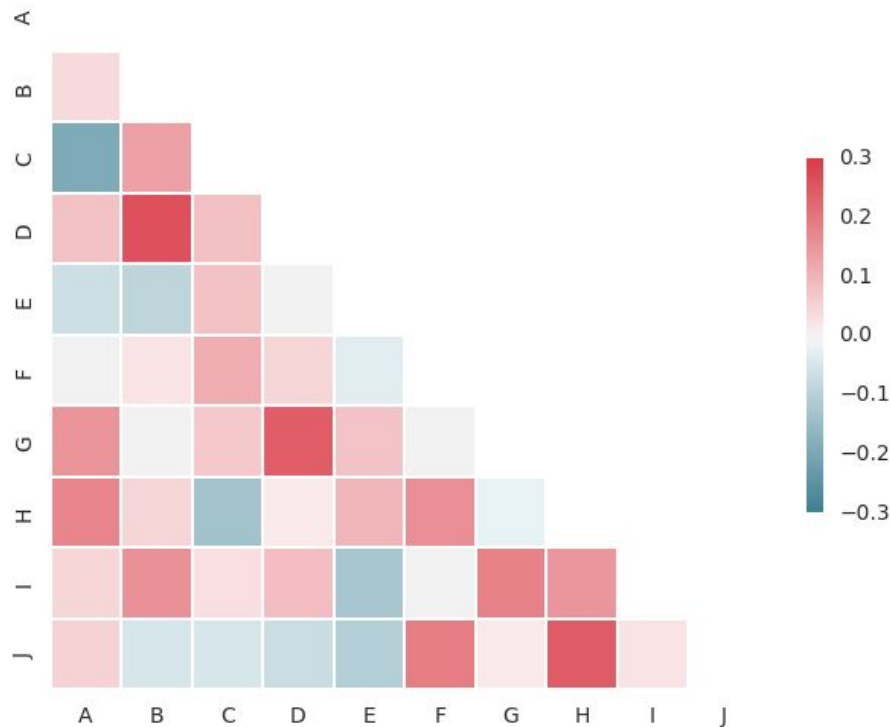
```
titles = {
    "Mme": "Mrs",
    "Ms": "Mrs",
    "Mrs": "Mrs",
    "Countess": "Royalty",
    "Lady": "Royalty"
}
```

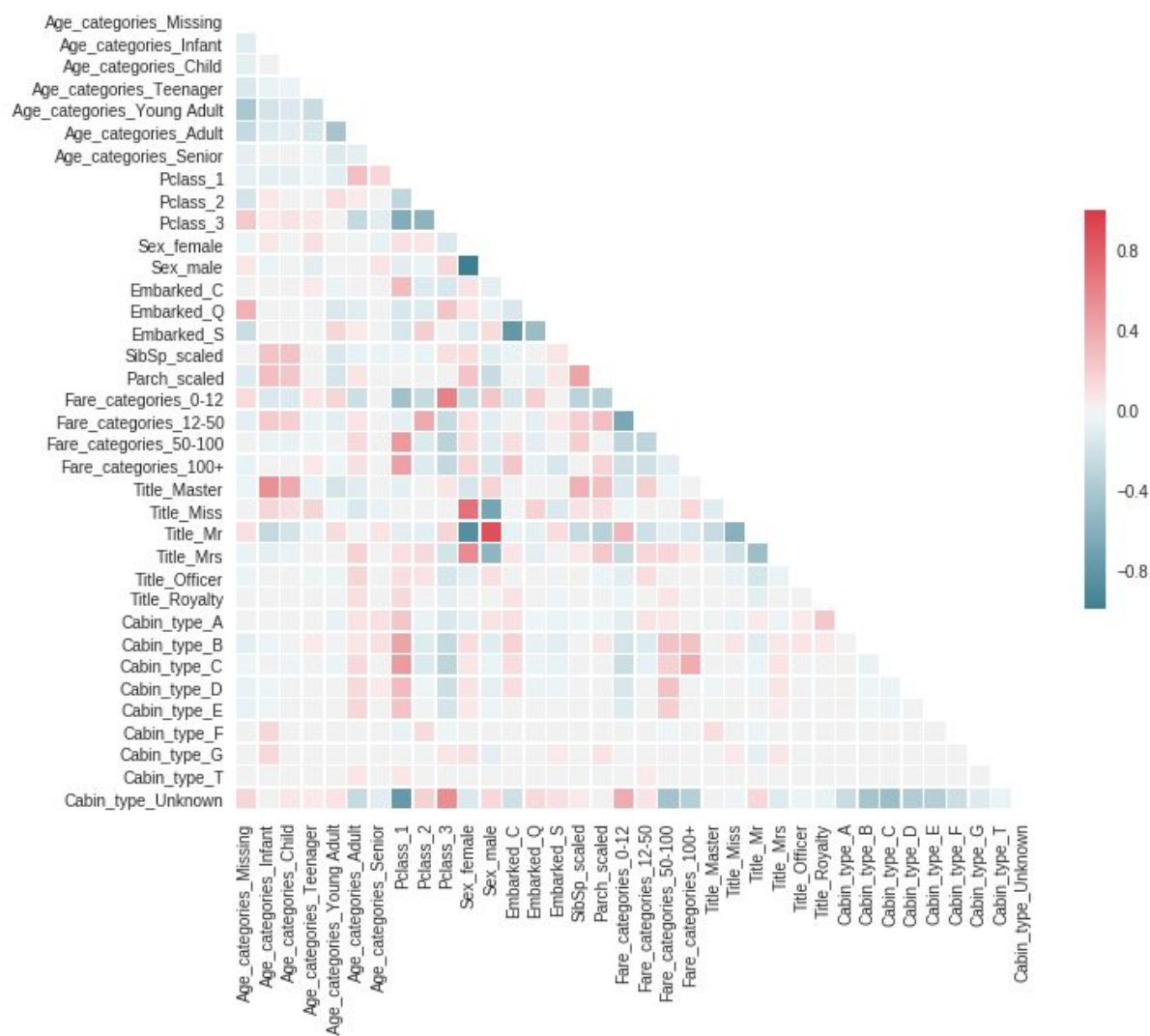
```
extracted_titles = train["Name"].str.extract('([A-Za-z]+\.)', expand=False)
train["Title"] = extracted_titles.map(titles)
```

	Name	Cabin
772	Mack, Mrs. (Mary)	E77
148	Navratil, Mr. Michel ("Louis M Hoffman")	F2
707	Calderhead, Mr. Edward Pennington	E24
879	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	C50
21	Beesley, Mr. Lawrence	D56
456	Millet, Mr. Francis Davis	E38
97	Greenfield, Mr. William Bertram	D10 D12
263	Harrison, Mr. William	B94
393	Newell, Miss. Marjorie	D36
759	Roths, the Countess. of (Lucy Noel Martha Dye...	B77

Finding Correlated Features

Collinearity














Finding Correlated Features

Final Feature Selection using RFECV

```
1 from sklearn.feature_selection import RFECV
2
3 columns = ['Age_categories_Missing', 'Age_categories_Infant',
4            'Age_categories_Child', 'Age_categories_Young Adult',
5            'Age_categories_Adult', 'Age_categories_Senior', 'Pclass_1', 'Pclass_3',
6            'Embarked_C', 'Embarked_Q', 'Embarked_S', 'SibSp_scaled',
7            'Parch_scaled', 'Fare_categories_0-12', 'Fare_categories_50-100',
8            'Fare_categories_100+', 'Title_Miss', 'Title_Mr', 'Title_Mrs',
9            'Title_Officer', 'Title_Royalty', 'Cabin_type_B', 'Cabin_type_C',
10           'Cabin_type_D', 'Cabin_type_E', 'Cabin_type_F', 'Cabin_type_G',
11           'Cabin_type_T', 'Cabin_type_Unknown']
12
13 all_X = train[columns]
14 all_y = train["Survived"]
15 lr = LogisticRegression()
16 selector = RFECV(lr,cv=10)
17 selector.fit(all_X,all_y)
18
19 optimized_columns = all_X.columns[selector.support_]
```

Training a model using our optimized columns

4012	new	Lech Brzozowski		0.78468	5	12h
4013	new	chen 10		0.78468	6	10h
4014	new	warrior2005		0.78468	3	10h
4015	new	Manoj reddy		0.78468	1	10h
4016	new	CharlieStElmo		0.78468	17	8h
4017	▲ 752	Topornin Dmitry		0.78468	8	7h
4018	▲ 1072	SuryaveerSingh		0.78468	8	4h
4019	new	Monster Zhong		0.78468	2	1h
4020	new	IvanovitchSilva		0.78468	3	~10s

Your Best Entry ↑

You advanced 1,983 places on the leaderboard!

Your submission scored 0.78468, which is an improvement of your previous score of 0.77033. Great job!



Tweet this!



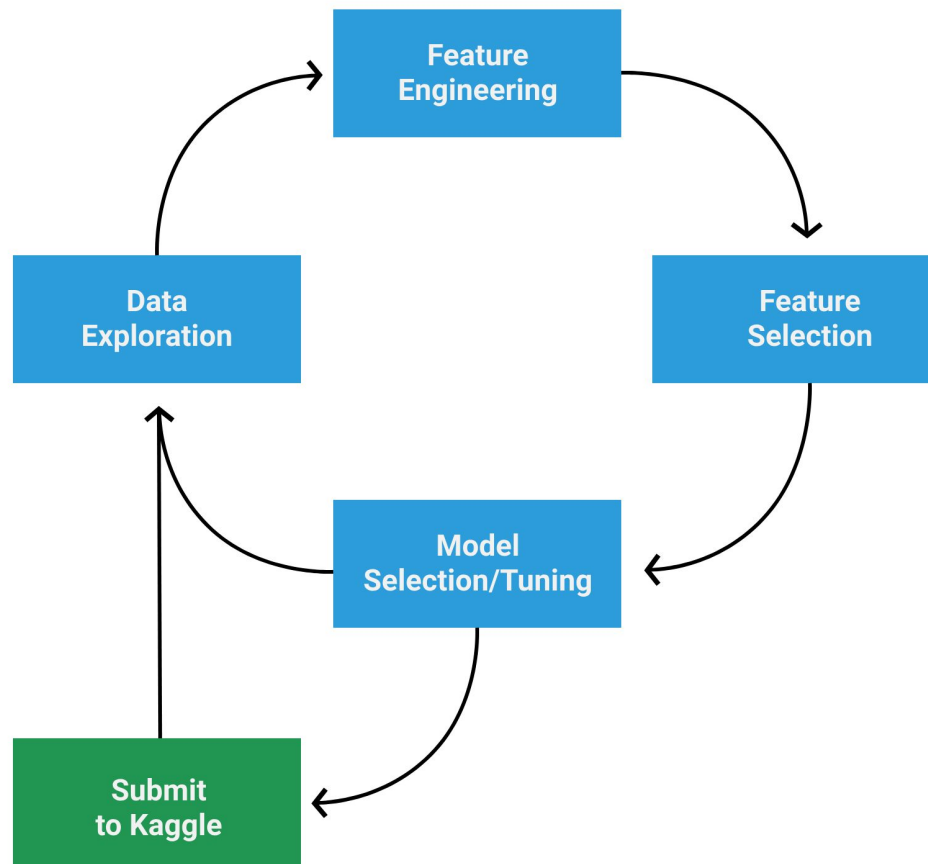
Next Steps

Here are some ideas that you can use to work with features for this competition:

- Use **SibSp** and **Parch** to explore total relatives onboard.
- Create combinations of multiple columns, for instance **Pclass + Sex**.
- See if you can extract useful data out of the **Ticket** column.
- Try different combinations of features to see if you can identify features that overfit less than others.
- we'll look at selecting and **optimizing different models** to improve our score

2. Feature Preparation, Selection and Engineering.ipynb





Model	Cross-validation score	Kaggle score
Previous best Kaggle score	82.36%	78.48%
Logistic regression baseline	82.38%	
K-nearest neighbors, k == 1	78.57%	
K-nearest neighbors, k == 19	82.38%	
K-nearest neighbors, best model from grid search	82.82%	75.59%
Random forests, default hyperparameters	80.70%	
Random forests, best model from grid search	84.28%	77.55%

3. Model Selection and Tuning.ipynb

4. Guided Project - Creating a Kaggle Workflow.ipynb

