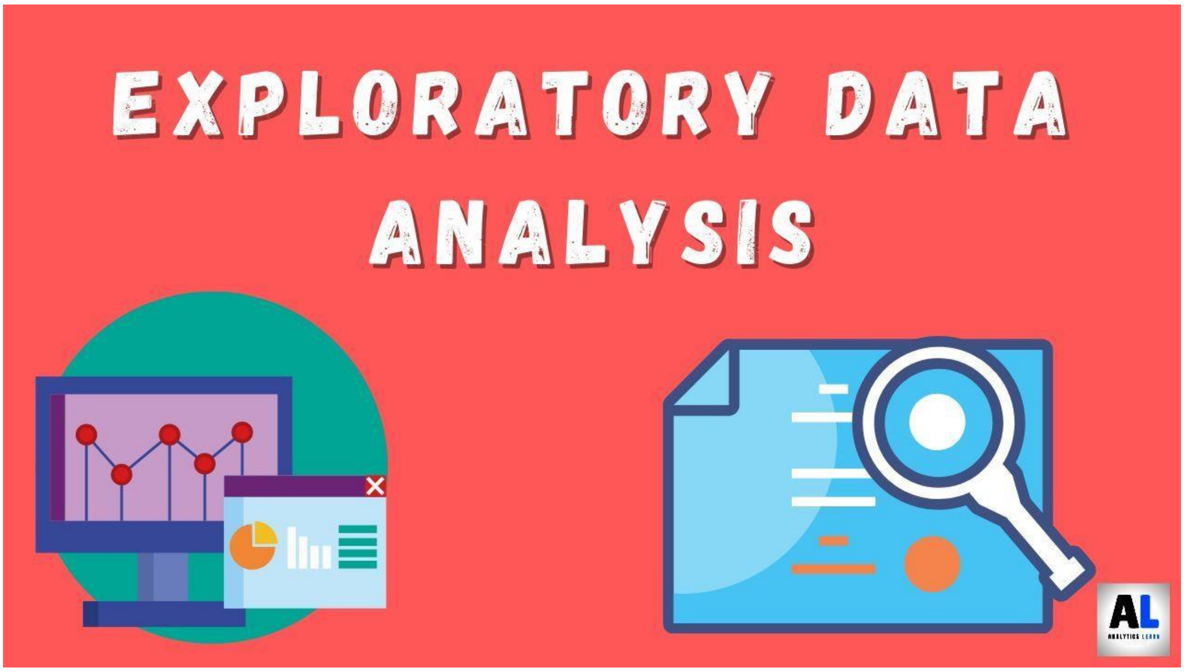
# Exploratorydataanalysisandfeatureextractionwith Python

Using**datavisualization,featureengineering** and**featureselection** tomakeasimple**logisticregression** lookpowerful!

**PHASE-4:DOCUMENTATION**




## 0. Belfast,anearlierincubator

Incubators are companies that support the creation of startups and their first years of activity. Theyareimportantbecausetheyhelpentrepreneurssolvesomeissuescommonlyassociated with running a business, such as workspace, training, and seed funding.

Ourengineeringmasterpiecealsoneedsastartingpoint.Inthissection,westarttheassemblage of our work by importing some libraries and general functions.

### Imports

In[1]:

linkcode

```
# Import libraries
import pandas as pd
import numpy as np
importseabornassns
import matplotlib.pyplot as plt
%matplotlib inline

# Put this when it's called
```

```
fromsklearn.model_selectionimporttrain_test_split
from sklearn.model_selection import learning_curve
fromsklearn.model_selectionimportvalidation_curve
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
```

# Functions

In[2]:

```
# Create table for missing data analysis
def draw_missing_data_table(df):
    total= df.isnull().sum().sort_values(ascending=False)
    percent=(df.isnull().sum()/df.isnull().count()).sort_values(ascending=False)
    missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
    return missing_data
```

In[3]:

```
# Plot learning curve
defplot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=1, train_sizes=np.linspace(.1, 1.0, 5)):
    plt.figure()
    plt.title(title)
    ifylimisnotNone:
        plt.ylim(*ylim)
    plt.xlabel("Trainingexamples")
    plt.ylabel("Score")
    train_sizes,train_scores, test_scores = learning_curve(
        estimator,X,y,cv=cv,n_jobs=n_jobs,train_sizes=train_sizes) train_scores_mean
    = np.mean(train_scores, axis=1)
    train_scores_std=np.std(train_scores,axis=1)
    test_scores_mean=np.mean(test_scores,axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()

    plt.fill_between(train_sizes,train_scores_mean-train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean+test_scores_std,alpha=0.1,color="g")
    plt.plot(train_sizes,train_scores_mean,'o-',color="r", label="Training
             score")
    plt.plot(train_sizes,test_scores_mean,'o-',color="g",
             label="Validation score")

    plt.legend(loc="best")
    return plt
```

In[4]:

```
# Plot validation curve
defplot_validation_curve(estimator,title,X,y,param_name,param_range,ylim=No ne,
cv=None,
                        n_jobs=1, train_sizes=np.linspace(.1, 1.0, 5)):
    train_scores,test_scores=validation_curve(estimator,X,y,param_name,para
m_range, cv)
```

```
    train_mean=np.mean(train_scores,axis=1)
    train_std = np.std(train_scores, axis=1)
    test_mean = np.mean(test_scores, axis=1)
    test_std = np.std(test_scores, axis=1)
    plt.plot(param_range,train_mean,color='r',marker='o',markersize=5,label='
Training score')
    plt.fill_between(param_range,train_mean+train_std,train_mean-train_std,
alpha=0.15, color='r')
    plt.plot(param_range,test_mean,color='g',linestyle='--',marker='s',marker
size=5, label='Validation score')
    plt.fill_between(param_range,test_mean+test_std,test_mean-test_std,alph a=0.15,
color='g')
    plt.grid()
    plt.xscale('log')
    plt.legend(loc='best')
    plt.xlabel('Parameter')
    plt.ylabel('Score')
    plt.ylim(ylim)
```

# 1. Theleandataset

linkcode

In the book 'The Lean Startup', Eric Ries tells us his personal experiences adapting lean management principles to high-tech startup companies. Through a series of anecdotes and stories,Riesteaches usallweneedtoknowaboutagilityandleanmethodologyinthestartup world.

Whileasetofimportantprinciplesaretaughtthroughoutthebook,thetruthisthatthelean startup methodology always ends up in an attempt to answer to the question: 'Should this product be built?'

To answer this question, the lean startup approach relies on a Build-Measure-Learn process. Thisprocessemphasizesrapiditerationasacriticalingredienttoproductdevelopment.Itgoes through the following phases:

1. **Build**.Figureouttheproblemthatneedstobesolved,generateideasabouthowtosolve it, and select the best one. Turn your best idea into a Minimum Viable Product (MVP).
2. **Measure**.Testyourproduct.Gotoyourcustomersandmeasuretheirreactionsand behaviors against your product.
3. **Learn**. Analyse the data you collected when testing the product with your customers. Drawconclusionsfromtheexperiment and decide whattodonext.Inother words,this is avalidatedlearningprocessthatquicklybuilds,tests,andrebuildsproducts,accordingto users' feedback. This reduces your market risks by failing fast and cheap, to get you closer and closer to what the market really needs.

Thiskerneldoessomethingsimilar.Wewilltrytofailfastandcheapbyquicklybuildingaworking end-to-end pipeline (Build). Then, we will instrument the system to evaluate its performance (Measure). Finally, we will make incremental changes to improve the system's performance (Learn). Note that this practical methodology was adapted from Goodfellow et al. (2016), a book you can access for free here.

Initially,wewillnotinvestmuchtimewiththeexploratorydataanalysis.Wewilljustdotheminimum viable effort to implement a reasonable model. This model will be our 'Minimum Viable Model'. Later, we will try to beat this model by enriching our data.

**Popquiz:**wasthe TitanicaMVP?

## 1.1. Doingthepitch

Startups use pitches to sell their idea. Accordingly, their pitch should be clear and concise, answeringquestionssuchas'whatdoyoudo?','whatdoyouwant?',and'who'sonyourteam?'. The pitch is important because investors are more willing to invest when they understand what you're doing.

Let'sreturnthefirstrowsofourdatasettogetaclearandconcisepictureofwhatisthereand what we can do with it.

In[5]:

```
# Import data
df= pd.read_csv('../input/train.csv')
df_raw = df.copy()# Save original data set, just in case.
```

In[6]:

```
# Overview
df.head()
```

Out[6]:

| | 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 |

Definitionsandquickthoughts:

- **PassengerId**.Uniqueidentificationofthepassenger.Itshouldn'tbenecessaryforthe machine learning model.
- **Survived**.Survival(0=No,1=Yes).Binaryvariablethatwillbeourtargetvariable.
- **Pclass**.Ticketclass(1=1st,2=2nd,3=3rd).Readytogo.
- **Name**.Nameofthepassenger.Weneedtoparsebeforeusingit.
- **Sex**.Sex.Categoricalvariablethatshouldbe encoded.
- **Age**.Agein years.Readyto go.
- **SibSp**.#ofsiblings/spousesaboardthe Titanic.Readytogo.
- **Parch**.#of parents/childrenaboardtheTitanic.Readyto go.
- **Ticket**.Ticketnumber.Bigmess.Weneedtounderstanditsstructurefirst.
- **Fare**.Passengerfare.Readytogo.
- **Cabin**.Cabinnumber.Itneedstobeparsed.
- **Embarked**.PortofEmbarkation(C=Cherbourg,Q=Queenstown,S=Southampton). Categorical feature that should be encoded.

The main conclusion is that we already have a set of features that we can easily use in our machinelearningmodel.Otherfeatures,like'Name','Ticket',and'Fare',requireanadditional effort before we can integrate them.

## 1.2. Showingthenumbers

Numbersarecrucialtosetgoals,tomakesoundbusinessdecisions,andtoobtainmoneyfrom investors. With numbers you can project the future of your startup, so that everyone can understand which are the expectations around your idea.

Inthesameway,wewillgeneratethedescriptivestatisticstogetthebasicquantitative information about the features of our data set.

In[7]:

```
# Descriptive statistics
df.describe()
```

Out[7]:

| | PassengerId | Survived | Pclass | Age | SibSp | Parch | Fare |
|---|---|---|---|---|---|---|---|
| count | 891.000000 | 891.000000 | 891.000000 | 714.000000 | 891.000000 | 891.000000 | 891.000000 |
| mean | 446.000000 | 0.383838 | 2.308642 | 29.699118 | 0.523008 | 0.381594 | 32.204208 |
| std | 257.353842 | 0.486592 | 0.836071 | 14.526497 | 1.102743 | 0.806057 | 49.693429 |
| min | 1.000000 | 0.000000 | 1.000000 | 0.420000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 223.500000 | 0.000000 | 2.000000 | 20.125000 | 0.000000 | 0.000000 | 7.910400 |
| 50% | 446.000000 | 0.000000 | 3.000000 | 28.000000 | 0.000000 | 0.000000 | 14.454200 |
| 75% | 668.500000 | 1.000000 | 3.000000 | 38.000000 | 1.000000 | 0.000000 | 31.000000 |
| max | 891.000000 | 1.000000 | 3.000000 | 80.000000 | 8.000000 | 6.000000 | 512.329200 |

ThereareThreeaspectsthatusuallycatchmyattentionwhenIanalysedescriptivestatistics:

1. **Minandmaxvalues**.Thiscangiveusanideaabouttherangeofvaluesandishelpful to detect outliers. In our case, all the min and max values seem reasonable and in a reasonable range of values. The only exception could eventually be the max value of 'Fare', but for now we will leave it as it is.
2. **Mean and standard deviation**. The mean shows us the central tendency of the distribution, while the standard deviation quantifies its amount of variation. For example, alowstandarddeviationsuggeststhatdatapointstendtobeclosetothemean.Givinga quick look to our values, there's nothing that looks like obviously wrong.
3. **Count**.Thisisimportanttogiveusafirstperceptionaboutthevolumeofmissingdata. Here, we can see that some 'Age' data is missing.

Sincethere'snothing shockingaboutthevariables,let'sproceedtothenextstep:missingdata.

## 1.3. Fillingthegaps

OneofmyfavouritedefinitionsofstartupbelongstoEricRies:'astartupisahumaninstitution designed to create a new product or service under conditions of extreme uncertainty'.

The word 'uncertainty' is key in this definition and it's also key in missing data. Missing data occurs when no data value on one or more variables is available. Consequently, it reduces the sizeofthedatasetandisapossiblesourceofbias,sincesomenon-randommechanismcanbe generating the missing data. As a result, missing data introduces uncertainty in our analysis.

Thereareseveralstrategiestodealwithmissingdata.Someofthemostcommonare:

- Useonlyvaliddata,deletingthecaseswheredatais missing.
- Imputedatausingvaluesfromsimilarcasesorusingthemeanvalue.
- Imputedatausingmodel-basedmethods,inwhichmodelsaredefinedto predictthe missing values.

Until today, I've never found a 'one size fits all' solution. I have some dogmas (e.g. I usually excludevariableswithmorethan25%of missingdata),butwhatusuallyguidesmyanalysisis intuition,criticalthinkingandneed (sometimesweneedtoleaveourdogmasat thedoor,ifwe want to generate some results).

My practical advice to handle missing data is to learn a different set of tools. Play with them accordingtoyourneeds,testthemand youshouldbefine.Agoodintroductiontothesubjectcan be found in Hair et al. (2013). This book has a practical summary about missing data and provides a framework that you can apply in almost all situations. Also, I wrote a technical paper comparing different imputation techniques, which I can share with you if you want.

Nowthatwecanseethetipoftheiceberg,let'sdiveintothesubject.

In[8]:

```
# Analyse missing data
draw_missing_data_table(df)
```

Out[8]:

| | Total | Percent |
|---|---|---|
| Cabin | 687 | 0.771044 |
| Age | 177 | 0.198653 |
| Embarked | 2 | 0.002245 |
| Fare | 0 | 0.000000 |
| Ticket | 0 | 0.000000 |
| Parch | 0 | 0.000000 |
| SibSp | 0 | 0.000000 |
| Sex | 0 | 0.000000 |
| Name | 0 | 0.000000 |
| Pclass | 0 | 0.000000 |
| Survived | 0 | 0.000000 |
| PassengerId | 0 | 0.000000 |

Firstthoughts:

- 'Cabin'hastoomanymissingvalues(>25%).Dogma!Weneedtodeletethisvariable right away.
- 'Age'canbeimputed.Fornow,I'llassociateavaluethatallowsmetoknowthatI'm imputing data. Later, I'll revise this strategy.
- Duetothelowpercentageofmissingvalues,I'lldeletetheobservationswherewedon't know 'Embarked'.

In[9]:

```
# Drop Cabin
df.drop('Cabin',axis=1,inplace=True) df.head()
```

Out[9]:

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

```
# Fill missing values in Age with a specific value
value = 1000
df['Age'].fillna(1000,inplace=True)
df['Age'].max()
```

Out[10]:

```
1000.0
```

In[11]:

linkcode

```
# Delete observations without Embarked
df.drop(df[pd.isnull(df['Embarked'])].index,inplace=True)#Getindexofpoints where Embarked is null
df[pd.isnull(df['Embarked'])]
```

Out[11]:

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|

## 1.4. Minimumviablemodel

The 'Minimum Viable Product' (MVP) is a key concept for any lean startup. Once the problem to solveisfiguredout,thefocusofthestartupshouldbeinthedevelopmentofasolution,theMVP, as fast as they can. Thanks to the MVP, it is possible to begin the learning process and improve the solution towards users needs.

Goodfellowetal.(2016)proposesananalogousapproachfor theapplicationofmachinelearning models.Astheauthorspointout,thesuccessfulapplicationofmachinelearningtechniquesgoes beyond the knowledge of algorithms and their principles. To successfully apply machine learning techniques, we need to start with a simple model that we can master and understand. Only then we should move to more complex algorithms.

Theauthorsproposeapracticalfour-stepsmethodology:

1. Selectaperformancemetricandatargetvalueforthismetric.Thismetricwillguideyour work and allow you to know how well you're performing. In our case, our performance metric will be 'accuracy' because it is the one defined by Kaggle.
2. Quicklysetupaworkingend-to-endpipeline.Thisshouldallowyoutoestimatethe selected performance metric.
3. Monitorthesystemtounderstanditsbehaviour,inparticulartounderstandwhetherits poor performance is related to underfitting, overfitting or defects.
4. Improve the system by iteration. Here we can apply feature engineering, tune hyperparametersorevenchangethealgorithm,accordingtotheoutputsofour monitoring system.

We will follow this methodology. Accordingly, our aim will be to get an initial model that we can use as a first baseline approach. This model will be our 'Minimum Viable Model' (MVM). Note thatrightnowitdoesn'tmattermuchhowwellthemodelperforms.Wejustneedastartingpoint. All in all, we're entrepreneurs. Worst case scenario, we name this model as 'beta version' :P

Ok,let'spreparethedatafortheMVMlaunching,fitalogisticregressiontoit,andanalysethe performance of the model through learning and validation curves.

### 1.4.1. Preparingthedata

In[12]:

```
# Data types
df.dtypes
```

Out[12]:

```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Embarked        object
dtype: object
```

- Wedon'tneed'PassengerId'forpredictionpurposes,sowewillexcludeit.
- 'Sex','Embarked',and'Pclass'shouldbecategorical.I'llnotconsider'Survived'as categorical because it's the output variable.
- Weneedtoparse'Name'and'Ticket'.Fornow,I'llignorethesefeatures.
- 'SibSp'couldbegroupedwith'Parch'tocreatea'Family'feature.Fornow,I'lljustidentify if the passenger is travelling alone or with family.

```
In[13]:
# Drop PassengerId
df.drop('PassengerId',axis=1,inplace=True)
df.head()
```

Out[13]:

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

```
# Define categorical variables
df['Sex'] = pd.Categorical(df['Sex'])
df['Embarked']=pd.Categorical(df['Embarked'])
```

```
In[15]:
# Create Family feature
df['FamilySize']=df['SibSp']+df['Parch']
df.head()
```

Out[15]:

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

```
# Drop SibSp and Parch
df.drop('SibSp',axis=1,inplace=True)
df.drop('Parch',axis=1,inplace=True)
df.head()
# Drop Name and Ticket
df.drop('Name',axis=1,inplace=True)
df.drop('Ticket',axis=1,inplace=True)
df.head()
```

Out[17]:

| | Survived | Pclass | Sex | Age | Fare | Embarked | FamilySize |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 3 | male | 22.0 | 7.2500 | S | 1 |
| 1 | 1 | 1 | female | 38.0 | 71.2833 | C | 1 |
| 2 | 1 | 3 | female | 26.0 | 7.9250 | S | 0 |
| 3 | 1 | 1 | female | 35.0 | 53.1000 | S | 1 |
| 4 | 0 | 3 | male | 35.0 | 8.0500 | S | 0 |

### 1.4.2. Launchingthemodel

Ready...Set...Go!

In[18]:

```
# Transform categorical variables into dummy variables
df= pd.get_dummies(df, drop_first=True)# To avoid dummytrap
df.head()
```

Out[18]:

| | Survived | Pclass | Age | Fare | FamilySize | Sex_male | Embarked_Q | Embarked_S |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 3 | 22.0 | 7.2500 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 38.0 | 71.2833 | 1 | 0 | 0 | 0 |
| 2 | 1 | 3 | 26.0 | 7.9250 | 0 | 0 | 0 | 1 |
| 3 | 1 | 1 | 35.0 | 53.1000 | 1 | 0 | 0 | 1 |
| 4 | 0 | 3 | 35.0 | 8.0500 | 0 | 1 | 0 | 1 |

```
# Create data set to train data imputation methods
X=df[df.loc[:,df.columns!='Survived'].columns] y =
df['Survived']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.2,random_sta te=1)
```

In[20]:

```
linkcode
# Debug
print('Inputs: \n', X_train.head())
print('Outputs:\n', y_train.head())
```

```
Inputs:
      Pclass     Age      FareFamilySizeSex_maleEmbarked_QEmbarked_S
121       31000.0      8.0500          0         1           0          1
```

```
687        3    19.0   10.1708           0         1           0           1
790        3  1000.0    7.7500           0         1           1           0
837        3  1000.0    8.0500           0         1           0           1
659        1    58.0  113.2750           2         1           0           0
Outputs:
 121     0
687    0
790    0
837    0
659    0
Name: Survived,dtype:int64
```

In[21]:

```
# Fit logistic regression
logreg=LogisticRegression()
logreg.fit(X_train, y_train)
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol   ver   to
silence this warning.
  FutureWarning)
```

Out[21]:

```
LogisticRegression(C=1.0,class_weight=None,dual=False,fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None,solver='warn',tol=0.0001,verbose=0,
                   warm_start=False)
```

In[22]:

```
# Model performance
scores = cross_val_score(logreg, X_train, y_train, cv=10)
print('CVaccuracy: %.3f +/- %.3f' % (np.mean(scores), np.std(scores)))
```

```
CVaccuracy:0.786+/-0.026
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol   ver   to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol   ver   to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol   ver   to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol   ver   to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specify  asol  ver  to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol   ver   to
silence this warning.
FutureWarning)
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
```

### 1.4.3. Assessingmodelperformance

In[23]:

```
# Plot learning curves
title="LearningCurves(LogisticRegression)" cv =
10
plot_learning_curve(logreg,title,X_train,y_train,ylim=(0.7,1.01),cv=cv,n_jo bs=1);
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
```


Validationcurvesinanutshell:

- Validationcurvesareatoolthatwecanusetoimprovetheperformanceofourmodel.It counts as a way of tuning our hyperparameters.
- They are different from the learning curves. Here, the goal is to see how the model parameterimpactsthetrainingandvalidationscores.Thisallowustochooseadifferent value for the parameter, to improve the model.
- Onceagain, ifthere isagapbetweenthetrainingandthevalidationscore,themodelis probablyoverfitting.Incontrast,ifthereisnogapbutthescorevalueislow,wecansay that the model underfits.

# 2. Thechubbydataset

Atthispoint,ourmodel:

- Canachievea0.786+/-0.026accuracy.
- Isbasedonalogistic regression.
- Uses'Pclass','Age','Fare','FamilySize','Sex',and'Embarkedasinputs;and'Survived' as output.

Moreover,concerningthepracticalmethodologythatwementionedbefore,wecansaythat:

1. Thechoiceoftheperformancemetricisaclosedtopicbecausewe'refollowingKaggle's specifications.
2. Ourcurrentmodelcanworkasabaselinemodelandresultedfromaworkingend-to-end pipeline.
3. Thelearningandvalidationcurvesallowustomonitorsystem's performance.

Asaconsequence,onlythefourthandlaststepofthepracticalmethodologyismissing:to improve the model by iteration. This can be done by:

- Improvingthewayhowwehandled'Age'missingdata.Inourleanapproachwedecided to replace missing data by a unique value, but now we can go deeper and search for a better imputation strategy.
- Exploringdatatounderstandwhichfeaturescanhaveimpactinthemodelandhowthey can be manipulated to boost that impact.
- Buildingnewfeaturesthatcanincreasethepredictivepowerofourmodel.

This will lead us to a heavy data analysis process, which aims to improve model's performance justbythedataqualityside.Inotherwords,wewillnotchangeourlearningalgorithmneitherwe will try to improve its parameters. We will only try to improve the performance of our model by enriching our data.

Thatsaid, let'ssaygoodbyetotheleanapproachandwelcome thechubbyapproach!

```
In[25]:
#Restartdataset
df=df_raw.copy()
df.head()
```

Out[25]:

| | 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 | PC17599 | 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 |

```
In[26]:
# Family size feature
df['FamilySize']=df['SibSp']+df['Parch']
df.head()


# Drop SibSp and Parch
df.drop('SibSp',axis=1,inplace=True)
df.drop('Parch',axis=1,inplace=True)
df.head()
```

Out[27]:

| | PassengerId | Survived | Pclass | Name | Sex | Age | Ticket | Fare | Cabin | Embarked | FamilySize |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | Braund, Mr. Owen Harris | male | 22.0 | A/521171 | 7.2500 | NaN | S | 1 |
| 1 | 2 | 1 | 1 | Cumings, Mrs.John Bradley(FlorenceBriggsTh... | female | 38.0 | PC17599 | 71.2833 | C85 | C | 1 |
| 2 | 3 | 1 | 3 | Heikkinen,Miss. Laina | female | 26.0 | STON/O2.3101282 | 7.9250 | NaN | S | 0 |
| 3 | 4 | 1 | 1 | Futrelle,Mrs. Jacques Heath(LilyMay Peel) | female | 35.0 | 113803 | 53.1000 | C123 | S | 1 |
| 4 | 5 | 0 | 3 | Allen,Mr. WilliamHenry | male | 35.0 | 373450 | 8.0500 | NaN | S | 0 |

In[28]:

linkcode

```
# Drop irrelevant features
df.drop(['Name','Ticket','Cabin'],axis=1,inplace=True)
df.head()
```

Out[28]:

| | PassengerId | Survived | Pclass | Sex | Age | Fare | Embarked | FamilySize |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | male | 22.0 | 7.2500 | S | 1 |
| 1 | 2 | 1 | 1 | female | 38.0 | 71.2833 | C | 1 |
| 2 | 3 | 1 | 3 | female | 26.0 | 7.9250 | S | 0 |
| 3 | 4 | 1 | 1 | female | 35.0 | 53.1000 | S | 1 |
| 4 | 5 | 0 | 3 | male | 35.0 | 8.0500 | S | 0 |

## 2.1. Imputationof'Age'missingdata

Our initial approach to estimate 'Age' missing values was to fill with a placeholder value (1000). Thisallowedusto quicklyget acompletedataset,inwhichwas easytoidentifyimputedvalues. Sinceourgoalwastohaveaworkingend-to-endpipelineasfastaspossible,thisapproachwas ok. However, it has several limitations. For example, we are using unrealistic replacement values, which are out of range and distort data distribution. Accordingly, now that we are improving the model, it makes sense to develop a different imputation method.

One way to improve our imputation method is to estimate the missing values based on known relationships.Inourcase,wecandothisbyusingtheinformationinthevariable'Name'.Looking to 'Name' values, we can see person's name and title. Person's title is a relevant information to estimate ages. To give an example, we know that a person with the title 'Master' is someone under 13 years old, since 'a boy can be addressed as master only until age 12'. Therefore, employing the information in 'Name' we can improve our imputation method.

Thestepstoimplementthisnewimputationmethodare:

- Extracttitlesfrom'Name'.
- Plotafigurewithbothfeaturesandconfirmthatthereisaconnectionbetweentitlesand age.
- Foreachtitle,getpeople'saverageageanduseittofillmissingvalues. Let's

see how this work, before you start with sinking feelings.

In[29]:

```
# Inspect names
df_raw['Name'].unique()[:10]
```

Out[29]:

```
array(['Braund,Mr.OwenHarris',
       'Cumings,Mrs.JohnBradley(FlorenceBriggsThayer)',
```

```
       'Heikkinen,Miss.Laina',
       'Futrelle,Mrs.JacquesHeath(LilyMayPeel)',
       'Allen,Mr.WilliamHenry','Moran,Mr.James',
       'McCarthy,Mr.TimothyJ','Palsson,Master.GostaLeonard',
       'Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)',
       'Nasser, Mrs. Nicholas (Adele Achem)'], dtype=object)
```

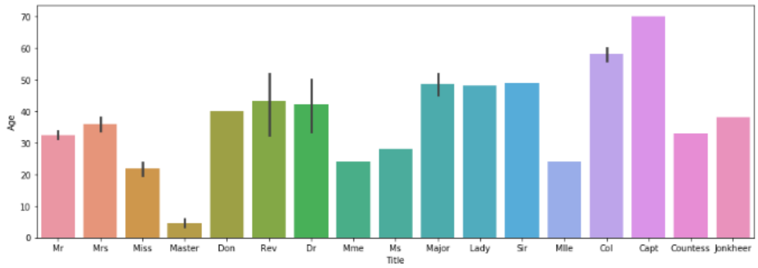
- Theruleseemsto be:*'lastname'+',' +'title'+'other names'*

```
In[30]:
# Extract titles from name
df['Title']=0
for i in df:
    df['Title']=df_raw['Name'].str.extract('([A-Za-z]+)\.',expand=False)#UseR EGEX
to define a search pattern
df.head()
Out[30]:
```

| | PassengerId | Survived | Pclass | Sex | Age | Fare | Embarked | FamilySize | Title |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | male | 22.0 | 7.2500 | S | 1 | Mr |
| 1 | 2 | 1 | 1 | female | 38.0 | 71.2833 | C | 1 | Mrs |
| 2 | 3 | 1 | 3 | female | 26.0 | 7.9250 | S | 0 | Miss |
| 3 | 4 | 1 | 1 | female | 35.0 | 53.1000 | S | 1 | Mrs |
| 4 | 5 | 0 | 3 | male | 35.0 | 8.0500 | S | 0 | Mr |

```
In[31]:

# Unique titles
df['Title'].unique()
array(['Mr','Mrs','Miss','Master','Don','Rev','Dr','Mme','Ms',
       'Major','Lady','Sir','Mlle','Col','Capt','Countess',
       'Jonkheer'], dtype=object)
In[32]:
linkcode
# Plot bar plot (titles, age and sex)
plt.figure(figsize=(15,5))
sns.barplot(x=df['Title'],y=df_raw['Age']);
```




- Bar plot gives us an estimate of central tendency for a numeric variable (height of each rectangle)andanindicationoftheuncertaintyaroundthatestimate(errorbarsinblack).
- Apart from Rev and Dr, which have a larger error bar, the mean value seems to accuratelyrepresentthedataofalltheotherfeatures.Thisvalidatesourapproach.
- Hereyoucanfindashortandsweetintrotoerrorbarsinterpretation.

```
In[33]:
# Means per title
df_raw['Title']=df['Title']#Tosimplifydatahandling
means=df_raw.groupby('Title')['Age'].mean() means.head()
```

```
Out[33]:
Title
Capt         70.0
Col          58.0
Countess     33.0
Don          40.0
Dr           42.0
Name:Age,  dtype:float64
```

```
In[34]:
# Transform means into a dictionary for future mapping
map_means=means.to_dict()
map_means
```

```
Out[34]:
{'Capt':70.0,
 'Col':58.0,
 'Countess':33.0,
 'Don':40.0,
 'Dr':42.0,
 'Jonkheer':38.0,
 'Lady':48.0,
 'Major':48.5,
 'Master':4.574166666666667,
 'Miss':21.773972602739725,
 'Mlle':24.0,
 'Mme':24.0,
 'Mr':32.368090452261306,
 'Mrs':35.898148148148145,
 'Ms': 28.0,
 'Rev':43.166666666666664,
 'Sir':49.0}
```

```
In[35]:
# Impute ages based on titles
idx_nan_age = df.loc[np.isnan(df['Age'])].index
```

```
df.loc[idx_nan_age,'Age'].loc[idx_nan_age]=df['Title'].loc[idx_nan_age].map(map_
means)
df.head()
```

Out[35]:

| | PassengerId | Survived | Pclass | Sex | Age | Fare | Embarked | FamilySize | Title |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | male | 22.0 | 7.2500 | S | 1 | Mr |
| 1 | 2 | 1 | 1 | female | 38.0 | 71.2833 | C | 1 | Mrs |
| 2 | 3 | 1 | 3 | female | 26.0 | 7.9250 | S | 0 | Miss |
| 3 | 4 | 1 | 1 | female | 35.0 | 53.1000 | S | 1 | Mrs |
| 4 | 5 | 0 | 3 | male | 35.0 | 8.0500 | S | 0 | Mr |

In[36]:

```
# Identify imputed data
df['Imputed'] = 0
df.at[idx_nan_age.values,'Imputed']=1
df.head()
```

Out[36]:

| | PassengerId | Survived | Pclass | Sex | Age | Fare | Embarked | FamilySize | Title | Imputed |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | male | 22.0 | 7.2500 | S | 1 | Mr | 0 |
| 1 | 2 | 1 | 1 | female | 38.0 | 71.2833 | C | 1 | Mrs | 0 |
| 2 | 3 | 1 | 3 | female | 26.0 | 7.9250 | S | 0 | Miss | 0 |
| 3 | 4 | 1 | 1 | female | 35.0 | 53.1000 | S | 1 | Mrs | 0 |

| | PassengerId | Survived | Pclass | Sex | Age | Fare | Embarked | FamilySize | Title | Imputed |
|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 5 | 0 | 3 | male | 35.0 | 8.0500 | S | 0 | Mr | 0 |

## 2.2. Exploratorydataanalysis

linkcode

Exploratory data analysis is often mentioned as one of the most important steps in the data analysis process. However, it's fairly easy to fall into a 'data diving' trap (especially if you're solvingproblemsaboutsunkenships)andgetlostintotheprocess.Whenthathappens,your analysis can end up like this.

We can avoid this by following a hypothesis driven approach. The hypothesis driven approach consists inestablishing hypothesis about thevariables behaviour and theirrelationships,earlyin theprocess,tothenfocusonusingdatatoprove(ordisprove)thosehypothesis.This makesour analysisveryobjectivebecausewewillbecollectingjustenoughdatatotestspecifichypothesis. As a result, we:

- Increasespeed.Sincewewilllimitouranalysistosomehypothesisandmoveforward.
- Reduceeffort.Theamountofdataandthenumberoftestswillbeonlywhatisneededto verify your hypothesis.
- Reducerisk.Ifyou'rerightyousucceedfast,ifyou'rewrongyoufailfast.

Here you can find one of my favourite PowerPoint presentations about the benefits and procedures of a hypothesis driven approach in problem solving. Please note that, particularly whenyoureallyneedtolearnaboutthedataset,itmakessensetoputthedivingcylinderandgo dive into the depths of data analysis. However, if at the outset you can generate an educated guess of what the answer of your problem is, I think that you should test your hypothesis and learn from it as fast as you can.

Ok, now that I convinced you that the hypothesis driven approach is the last coke in the desert, let me show you how toapplyit. Cases like the one we have are easytargets for the hypothesis driven approach because we don't have many variables, so we can more or less guess their impact. Accordingly, we will start by listing each of the variables and generate hypothesis about theirrelationshipwiththetargetvariable('Survived').Then,wewilltestthosehypothesisthrough a set of exploratory data analysis tools. As a result, we'll end up with a comprehensive view about the variables that should belong to our prediction model.

Let'sgetstarted:

- **PassengerId**.Thisisjustanuniqueidentificationofeachpassenger.It'snotexpectedto be relevant to our analysis.
- **Survived**.Targetvariable.Tosinkornottosinkisthequestionofthisexercise.
- **Pclass**.Thisistheticketclass.AccordingtoKarlMarx,thisshouldaffectourtarget variable. First class should have a higher survival rate.
- **Name**.Namesareaformofsocialtagging,especiallywhenaccompaniedbyatitle.Asa consequence, it can lead to different forms of treatment. Let's keep an eye on this.
- **Sex**.Alwaysimportant.

- **Age**.Itshouldmakeadifference.Forexample,childrenareusuallyevacuatedfirstina disaster, so that we can think about a solution in silence... Joking, the true reason why 'Age' matters is this one.
- **SibSp**.Numberofsiblings/spousesaboardtheTitanic.I'dsaythatit'seasiertosurviveif you're with your family than if you're travelling alone. Teamwork!
- **Parch**.Numberofparents/childrenaboardtheTitanic.Itshouldplaywith'SibSp'.
- **Ticket**.Thisistheticketnumber.Unlessithassomeinformationaboutplaces,it shouldn't be important for prediction purposes.
- **Fare**.Samelogicas'Pclass'.
- **Cabin**.Thecabinnumbercanindicatewherepeopleweereduringthedisaster.Itwouldn't be surprising if it had some influence in survival chances, but this variable was excluded due to the high percentage of missing values.
- **Embarked**. When the sun rises, it rises for everyone. It's not expectable that people coming from Cherbourg are more unlucky than people coming from Southampton.Unlessthereissomesecondordereffect, likerefusingtorunawaytokeepyourhonoras a man, I would say that this variable is not important.

Now,stepbystep,let'sperformouranalysis.

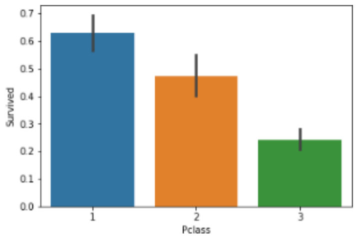
### 2.2.1. Pclass

Ourhypothesisisthatthehighertheclass,thehigherthechancesofsurvival.Thismeansthata person travelling in the first class has a higher chance of survival than a person traveling on the second or third class.

Tovisualizeifthereisarelationshipbetween'Pclass'and'Survival',let'sdoabarplot.

In[37]:

```
# Plot
sns.barplot(df['Pclass'],df['Survived']);
```




Aswecansee,about60%ofthepeopletravellinginthefirstclasssurvived.Incontrast,only approximately 25% of the people travelling in the third class survived. Accordingly, this plot suggests that the class in which people travel affects the chances of survival.

### 2.2.2. Name/Title

Our assumption is that people's title influences how they are treated. In our case, we have severaltitles,butonlysomeofthemaresharedbyasignificantnumberofpeople.Accordingly,it would be interesting if we could group some of the titles and simplify our analysis.

Let'sanalysethetitleandseeifwecancanfindasensiblewaytogroupthem.Then,wetestour new groups and, if it works in an acceptable way, we keep it. For now, optimization will not be a goal. The focus is on getting something that can improve our current situation.

In[38]:

```
# Count how many people have each of the titles
df.groupby(['Title'])['PassengerId'].count()
```

Out[38]:

```
Title
Capt          1
Col           2
Countess      1
Don           1
Dr            7
Jonkheer      1
Lady          1
Major         2
Master       40
Miss        182
Mlle          2
Mme           1
Mr          517
Mrs         125
Ms            1
Rev           6
Sir           1
Name:PassengerId,dtype:int64
```

Fromtheresultsabovewecanseethat:

- Titleslike'Master','Miss','Mr',and'Mrs',appearseveraltimes.Accordingly,wewillnot group them.
- RegardingMmeandMlle,wecansee herethattheycorrespondtothecategoriesMrs and Miss, respectively. As a consequence, we will assign them to those titles.
- Finally,wewillgroupalltheothertitlesinanewtitlenamed'Other'.Then,wewilldefine 'Title' as a categorical feature and plot it to see how it looks like. If it looks ok, we will proceed with this new categorization.

In[39]:

```
# Map of aggregated titles:
titles_dict = {'Capt': 'Other',
               'Major': 'Other',
               'Jonkheer': 'Other',
               'Don': 'Other',
               'Sir': 'Other',
               'Dr': 'Other',
               'Rev': 'Other',
               'Countess': 'Other',
               'Dona': 'Other',
               'Mme': 'Mrs',
               'Mlle': 'Miss',
               'Ms': 'Miss',
               'Mr': 'Mr',
               'Mrs': 'Mrs',
               'Miss': 'Miss',
               'Master': 'Master',
               'Lady': 'Other'}
```

In[40]:

```
# Group titles
df['Title']=df['Title'].map(titles_dict)
df['Title'].head()
0       Mr
1      Mrs
```

```
2     Miss
3      Mrs
4       Mr
Name:Title,dtype:object In
```

[41]:

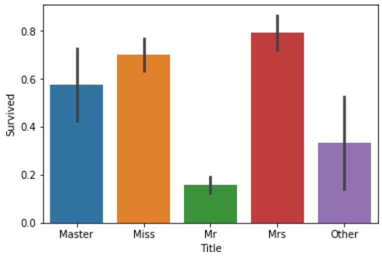
```
# Transform into categorical
df['Title']=pd.Categorical(df['Title'])
df.dtypes
```

Out[41]:

```
PassengerId       int64
Survivedint64
Pclass            int64
Sex              object
Age             float64
Fare            float64
Embarkedobject
FamilySizeint64Titlecat
egory Imputed     int64
dtype: object
```

In[42]:

```
# Plot
sns.barplot(x='Title',y='Survived', data=df);
```




Aswealreadyknow,thebarplotshows usanestimateofthemeanvalue(heightofeach rectangle) and an indication of the uncertainty around that central tendency (error bars).

Ourresultssuggestthat:

- Peoplewiththetitle'Mr'survivedlessthanpeoplewithanyothertitle.
- Titles withasurvivalratehigherthan50%arethosethatcorrespondtofemale(Missor Mrs) or children (Master) titles.
- Our new category, 'Other', should be more discretized. As we can see by the error bar (blackline),there isasignificant uncertaintyaroundthemeanvalue.Probably, one ofthe problems is that we are mixing male and female titles in the 'Other' category. We should proceed with a more detailed analysis to sort this out. Also, the category 'Master' seems tohaveasimilarproblem.Fornow,wewillnotmakeanychanges,butwewillkeepthese two situations in our mind for future improvement of our data set.

### 2.2.3. Sex¶

Sex is one of the most discussed topics in Human history. There are several perspective about the topic, but I must confess that Freud's perspectives had a significant impact on me because they have shown me the subject in a new perspective. What's new about Freud is that he associatedmany'normal'behaviourstosexualdrives,almosttothepointofmakingusquestion everything we do. In the end, Freud realized that not everything was about sex. As he said, 'sometimes a cigar is just a cigar' (Freud used to smoke cigars).

Preamblesaside,whatwereallyneedtoknowisifsometimesacigarisjustacigarornot.We alreadyhavesomeclues that,inTitanic, womenhad ahigher survivalrate.But, nothing better than a plot to see what's going on.

In[43]:

```
# Transform into categorical
df['Sex']= pd.Categorical(df['Sex'])
```

In[44]:

```
# Plot
sns.barplot(df['Sex'],df['Survived']);
```

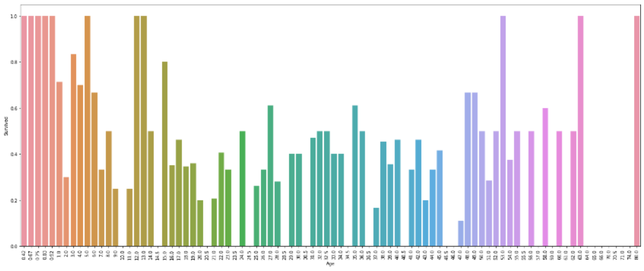
### 2.2.4. Age

'Age' is the next variable in the list. Our hypothesis is that children are more prone to survive, while people in its adult life may have a lower rate of survival. Personally, I don't have any specialintuitionaboutelders,sincetheyarethemostvulnerable.Thiscanplayforbothsides: either people helpelders becausetheyare morevulnerable, or theytheyare notable tocope with the challenges posed by the wreck of a ship.

Let'scallthe usualsuspect(barplot)tohelpusunderstandingthesituation.

In[45]:

```
# Plot
plt.figure(figsize=(25,10))
sns.barplot(df['Age'],df['Survived'],ci=None)
plt.xticks(rotation=90);
```



Withalittle bitofcreativity,wecansaythat theplothasthreeregions:

1. Oneregionthatgoes betweenage0and15;
2. Onebetweenage15and48;
3. Alastonebetweenage48and80.

I know that this division is arguable, especially in what concerns the last two categories. However,thepointisthatthiscategoriessplitfits intowhatweknowaboutthewayoursocietyis organized: childrens, adults and elders. For now, let's proceed this way.

In[46]:

```
#Plot
'''
Probably,thereisaneasierwaytodothisplot.Ihadaproblemusing plt.axvspan
because the xmin and xmax values weren't
beingplottedcorrectly.Forexample,Iwoulddefinexmax=12andonly
theareabetween0and7wouldbefilled.Thiswashappeningbecausemy X-axis don't
follow a regular (0, 1, ..., n) sequence. After some trial
anderror,Inoticedthatxminandxmaxrefertothenumberofelementsin  the  X-axis
coordinate that should be filled. Accordingly, I defined two
```

```
variables,x_limit_1andx_limit_2,thatcountthenumberofelementsthat should be
filled in each interval. Sounds confusing? To me too.
'''
limit_1 = 12
limit_2 = 50

x_limit_1=np.size(df[df['Age']<limit_1]['Age'].unique()) x_limit_2=
np.size(df[df['Age'] < limit_2]['Age'].unique())

plt.figure(figsize=(25,10))
sns.barplot(df['Age'],df['Survived'],ci=None)

plt.axvspan(-1, x_limit_1, alpha=0.25, color='green')
plt.axvspan(x_limit_1,x_limit_2,alpha=0.25,color='red')
plt.axvspan(x_limit_2, 100, alpha=0.25, color='yellow')

plt.xticks(rotation=90);
# Bin data
df['Age']=pd.cut(df['Age'],bins=[0,12,50,200],labels=['Child','Adult','Elde r'])
df['Age'].head()

Out[47]:
0     Adult
1     Adult
2     Adult
3     Adult
4     Adult
Name:Age,dtype:category
Categories(3,object):[Child<Adult<Elder] In
[48]:
# Plot
sns.barplot(df['Age'],df['Survived']);
```

Theplotshowsthatchildren haveahighersurvivalrate.Italsoshowsthat,intermsofsurvival, there is not a significant difference between the categories 'Adult' and 'Elder'. For now, we will not make any change because there is a theoretical rationale behind this categorization. Nonetheless,itseemsthatitwouldbeenoughtojustdistinguishbetweenchildrenandadults.

### 2.2.5. FamilySize

Regardingfamilysize,ourhypothesisisthatthosewhotravelalone,havealowersurvivalrate. The idea is that people with family can collaborate and help each other escaping.

Let'sseeifthatmakessenseusingourbeautifulandonlyfriend,thebarplot.

```
In[49]:

# Plot
sns.barplot(df['FamilySize'],df['Survived']);
```

As we can see, when 'FamilySize' is between 0 and 3, our hypothesis finds some support. Peoplethataretravellingalonehavealowersurvivalratethanpeoplewhoaretravellingwith one, two or three people more.

However, when FamilySize is between 4 and 10, things start to change. Despite the large variabilityoftheresults,thesurvivalratedrops.Thismaysuggestthatourhypothesisshouldbe revised when 'FamilySize' is higher than 3.

This variable seems to be more complex than expected. Accordingly, we will not make any transformationinthisvariableandwewillleaveitasacontinuousvariabletopreserveallthe information it has.

### 2.2.6. Fare

Thesamelogicappliedto'Pclass'shouldworkfor'Fare':higherfares,highersurvivalrate.

Sincenowwewanttoestablishcomparisonsacrossdifferentlevelsofacategoricalvariable,we will use a box plot instead of a bar plot.

In[50]:

```
# Plot
plt.figure(figsize=(7.5,5))
sns.boxplot(df['Survived'],df['Fare']);
```

Theplotsuggeststhatthosewhosurvivedpaidahigherfare.Sincewebelievethisvariableis connected with 'Pclass', let's see how they work together.

In[51]:

```
# Plot
sns.barplot(df['Survived'],df['Fare'], df['Pclass']);
```

Here we have an interesting result. It seems that 'Fare' doesn't make difference, in terms of survival,ifyouaretravellinginsecondorthirdclass.However,ifyouaretravellinginfirstclass, thehigherthefare,thehigherthechances ofsurvival.Consideringthis,itwouldmakesenseto create interaction features between 'Fare' and 'Pclass'.

### 2.2.7. Embarked

linkcode

Thehypothesisregarding'Embarked'isthatitdoesn'tinfluencethechancesofsurvival.Itishard toimagine ascenario in whichpeoplefrom Southampton,for instance, would suchacompetitive advantage, that it would make them more apt for survival than people from Queensland. Yes, inDarwinwebelieve.

Asimpleplotcanenlightenus.

In[52]:

```
# Plot
sns.barplot(df['Embarked'],df['Survived']);
# Compare with other variables
df.groupby(['Embarked']).mean()
```

Out[53]:

| | PassengerId | Survived | Pclass | Fare | FamilySize | Imputed |
|---|---|---|---|---|---|---|

| Embarked | | | | | | |
|---|---|---|---|---|---|---|
| C | 445.357143 | 0.553571 | 1.886905 | 59.954144 | 0.750000 | 0.226190 |
| Q | 417.896104 | 0.389610 | 2.909091 | 13.276030 | 0.597403 | 0.636364 |
| S | 449.527950 | 0.336957 | 2.350932 | 27.079812 | 0.984472 | 0.139752 |

ItseemsthatpeopleembarkingonCwerepayingmoreandtravellinginabetterclassthan people embarking on Q and S.

In[54]:
```
# Relationship with age
df.groupby(['Embarked','Age'])['PassengerId'].count()
```

Out[54]:
```
EmbarkedAge
C         Child     11
          Adult    104
          Elder     15
Q         Child      4
          Adult     21
          Elder      3
S         Child     54
          Adult    455
          Elder     45
Name:PassengerId,  dtype:int64
```
Nosignificantdifferencescanbefound.

In[55]:
```
# Relationship with sex
df.groupby(['Embarked','Sex'])['PassengerId'].count()
```

Out[55]:
```
EmbarkedSex
C         female     73
          male       95
Q         female     36
          male       41
S         female    203
          male      441
Name:PassengerId,  dtype:int64
```
Nosignificantdifferencescanbefound.

Consideringtheresultsabove,Ifeeltemptedtosaythattheembarkmentpointdoesn'tinfluence the survival rate. What really seems to be influencing is the class where people were travelling and how much they were spending.

Fornow,IwillnotdeletethevariablebecauseIfeelthatI'malittlebitbiasedandtryingtoforcea conclusion. However, let's keep in mind that maybe 'Embarked' doesn't affect 'Survived'.

## 2.3. Featureextraction

In the book 'How Google Works', Eric Schmidt and Jonathan Rosenberg refer that Google's secret sauce is 'technical insight'. According to the authors, it is fundamental technical insight thatallowscompaniestocreategreatproducts,whichproviderealvaluetothecustomers.For example, PageRank gave an incredible competitive advantage to Google in relation to other search engines, by providing a far better way to rank search results on the web.

Feature extraction is our technological insight in machine learning. It addresses the problem of attaining the most informative and compact set of features, to improve the performance of machine learning models. Let's go step-by-step. First, we are talking about 'informative'. This meansthatwearelookingforfeaturesthatcancharacterizethebehaviourofwhatwearetrying tomodel.Forinstance,ifwewanttomodeltheweather,features liketemperature,humidityand wind are informative (they are related to the problem). By contrast, the result of a football game will not be an informative feature because it doesn't affect the weather.

Regarding 'compact', what we mean is that we want to exclude irrelevant features from our model. There are several reasons to exclude irrelevant features. In our case, I'd say that the mostimportantistoreduceoverfitting.Takingtheweatherexampleagain:weknowthatfootball scores do not affect weather, but suppose that all rain instances in our training set happen to occur after a Benfica victory. Then, our model might learn that rain is related to Benfica's victories, which is not true. Such an incorrect generalization from an irrelevant feature of the training set would result in a machine learning model that fits a particular set of data, but fails to predict future observations reliably (overfitting).

Thesetwomainissuesareaddressedinthefollowingsub-sections:

1. Featureengineering,whichisrelatedtothegenerationofinformativefeatures;
2. Featureselection,whichregardsthechoiceofacompactsetoffeatures.

### 2.3.1. Featureengineering

Feature engineering is the art of converting raw data into useful features. There are several featureengineeringtechniquesthatyoucanapplytobeanartist.Acomprehensivelistofthemis presented by Heaton (2016). We will use just two techniques:

- Box-Coxtransformations(Box&Cox1964);
- Polynomialsgenerationthroughnon-linearexpansions.

Beforetheapplicationofthesetechniques,wewilljustmakesomeadjustmentstothedata,in order to prepare it for the modelling process.

*Datapreparation*

In[56]:

```
# Overview
df.head()
```

Out[56]:

| | PassengerId | Survived | Pclass | Sex | Age | Fare | Embarked | FamilySize | Title | Imputed |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 3 | male | Adult | 7.2500 | S | 1 | Mr | 0 |

| | PassengerId | Survived | Pclass | Sex | Age | Fare | Embarked | FamilySize | Title | Imputed |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 1 | female | Adult | 71.2833 | C | 1 | Mrs | 0 |
| 2 | 3 | 1 | 3 | female | Adult | 7.9250 | S | 0 | Miss | 0 |
| 3 | 4 | 1 | 1 | female | Adult | 53.1000 | S | 1 | Mrs | 0 |
| 4 | 5 | 0 | 3 | male | Adult | 8.0500 | S | 0 | Mr | 0 |

In[57]:

```
# Drop feature
df.drop('PassengerId',axis=1, inplace=True)
```

In[58]:

```
# Check features type
df.dtypes
```

Out[58]:

```
Survivedint64
Pclass             int64
Sex             category
Age             category
Fare             float64
Embarkedobject
FamilySizeint64Titleca
tegoryImputedint64dtyp
e: object
# Transform object into categorical
df['Embarked']=pd.Categorical(df['Embarked'])
df['Pclass'] = pd.Categorical(df['Pclass'])
df.dtypes
```

Out[59]:

```
Survivedint64Pclasscat
egory
Sex             category
Age             category
Fare             float64
Embarked        category
FamilySizeint64Titleca
tegoryImputedint64dtyp
e: object
```

```
In[60]:
# Transform categorical features into dummy variables
df=pd.get_dummies(df,drop_first=1)
df.head()
```

```
# Get training and test sets
fromsklearn.model_selection import train_test_split

X=df[df.loc[:,df.columns!='Survived'].columns] y =
df['Survived']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=.2,random_sta te=0)
```

*Box-Coxtransformations*

Box-Cox transformations aim to normalize variables. These transformations are an alternative to thetypicaltransformations,suchassquareroottransformations,logtransformations,andinverse transformations.ThemainadvantageofBox-Coxtransformationsisthattheyoptimallynormalize the chosen variable. Thus, they avoid the need to randomly try different transformations and automatize the data transformation process.

```
In[62]:

# Apply Box-Cox transformation
from scipy.statsimport boxcox

X_train_transformed = X_train.copy()
X_train_transformed['Fare']=boxcox(X_train_transformed['Fare']+1)[0]
X_test_transformed = X_test.copy()
X_test_transformed['Fare']= boxcox(X_test_transformed['Fare'] + 1)[0]
```

*Polynomials*

Onestandardwaytoenrichoursetoffeaturesistogeneratepolynomials.Polynomialexpansion creates interactions between features, as well as creates powers (e.g. square of a feature). This way we introduce a nonlinear dimension to our data set, which can improve the predictive power of our model.

Weshouldscaleourfeatureswhenwehavepolynomialorinteractiontermsinourmodel.These terms tend to produce multicollinearity, which can make our estimates very sensitive to minor changes in the model. Scaling features to a range allow us to reduce multicollinearity and its problems.

Toscalethefeatures, we will transformthe data so thatit lies between agivenminimumand maximumvalue.Wewillfollowthecommonpracticeandsaythatourminimumvalueiszero, and our maximum value is one.

```
In[63]:

# Rescale data
from sklearn.preprocessingimport MinMaxScaler

scaler = MinMaxScaler()
X_train_transformed_scaled=scaler.fit_transform(X_train_transformed)
X_test_transformed_scaled = scaler.transform(X_test_transformed)
In[64]:

# Get polynomial features
from sklearn.preprocessingimport PolynomialFeatures

poly =PolynomialFeatures(degree=2).fit(X_train_transformed)
```

```
X_train_poly=poly.transform(X_train_transformed_scaled)
X_test_poly = poly.transform(X_test_transformed_scaled)
```

In[65]:

```
# Debug
print(poly.get_feature_names())
['1','x0','x1','x2','x3','x4','x5','x6','x7','x8','x9','x10','x11'
,'x12','x13','x0^2','x0x1','x0x2','x0x3','x0x4','x0x5','x0x6',
'x0x7','x0x8','x0x9','x0x10','x0x11','x0x12','x0x13','x1^2','x1
x2','x1x3','x1x4','x1x5','x1x6','x1x7','x1x8','x1x9','x1x10',
'x1x11','x1x12','x1x13','x2^2','x2x3','x2x4','x2x5','x2x6','x2
x7','x2x8','x2x9','x2x10','x2x11','x2x12','x2x13','x3^2','x3 x4'
,'x3x5','x3x6','x3x7','x3x8','x3x9','x3x10','x3x11','x3x12','
x3x13','x4^2','x4x5','x4x6','x4x7','x4x8','x4x9','x4x10','x4x1
1','x4x12','x4x13','x5^2','x5x6','x5x7','x5x8','x5x9','x5x10',
'x5x11','x5x12','x5x13','x6^2','x6x7','x6x8','x6x9','x6x10','x6
x11','x6x12','x6x13','x7^2','x7x8','x7x9','x7x10','x7x11','x7x1
2','x7x13','x8^2','x8x9','x8x10','x8x11','x8x12','x8x13','x9^2',
'x9x10','x9x11','x9x12','x9x13','x10^2','x10x11','x10x12','x10x1
3','x11^2','x11x12','x11x13','x12^2','x12x13','x13^2']
```

### 2.3.2. Featureselection

The next step is to perform feature selection. Feature selection is about chosing the relevant information. It is good to add and generate features, but at some point we need to exclude irrelevantfeatures.Otherwise,wewillbepenalizingthepredictivepowerofourmodel.Youcan find a concise introduction to the feature selection subject in Guyon & Elisseeff (2003).

Inthiswork,wewilluseaunivariatestatisticsapproach.Thisapproachselectsfeaturesbased onunivariatestatisticaltestsbetweeneachfeatureandthetargetvariable.Theintuitionisthat features that are independent from the target variable, are irrelevant for classification.

We will use the chi-squared test for feature selection. This means that we have to choose the numberoffeaturesthatwewantinthemodel.Forexample,ifwewanttohavethreefeaturesin our model, the method will select the three features with highest $\chi 2$�2 score. Sincewedon'tknowtheidealnumberoffeatures,wewilltestthemethodwithallthepossible number of features and choose the number of features with better performance.

*Univariatestatistics*

In[66]:

```
# Select features using chi-squared test
fromsklearn.feature_selectionimportSelectKBest from
sklearn.feature_selection import chi2

##Getscoreusingoriginalmodel logreg =
LogisticRegression(C=1)
logreg.fit(X_train, y_train)
scores = cross_val_score(logreg, X_train, y_train, cv=10)
print('CVaccuracy(original):%.3f+/-%.3f'%(np.mean(scores),np.std(scores)))
highest_score = np.mean(scores)

## Get score using models with feature selection
for iin range(1, X_train_poly.shape[1]+1, 1):
    # Select i features
    select=SelectKBest(score_func=chi2,k=i)
    select.fit(X_train_poly, y_train)
```

```
    X_train_poly_selected= select.transform(X_train_poly)

    #Model with i features selected
    logreg.fit(X_train_poly_selected,y_train)
    scores=cross_val_score(logreg,X_train_poly_selected,y_train,cv=10) print('CV
    accuracy (number of features = %i): %.3f +/- %.3f' % (i,
                                                                      np.mean(score
s),
                                                                      np.std(scores
)))

    # Save results if best score
    ifnp.mean(scores)>highest_score:
        highest_score=np.mean(scores) std
        = np.std(scores)
        k_features_highest_score = i
    elifnp.mean(scores)==highest_score: if
        np.std(scores) < std:
            highest_score=np.mean(scores)
            std = np.std(scores)
            k_features_highest_score = i

# Print the number of features
print('Numberof features when highest score: %i' % k_features_highest_score)
```

# 3. Unicornmodel

Startupsusetheterm'unicorn'to describe astartupthatisvaluedat one billiondollars ormore. RegardlessofwhetheritisaEuropeanorAmericanbillion,onebillionisabignumber.Actually, it is so big and rare that when we find startups with such value, we associate them to those mythical creatures that are the unicorns.

We'vebeenthroughalongjourneysincewestartedsolvingthisproblem.Sinceourstartwitha lean model, we've been scaling our startup: we imputed missing data, we performed an exploratory data analysis and we extracted features. We also had to deal with terrible Titanic jokes that take some time to sink in.

Nowit'stimetoturnallthisworkintoahighlyaccuratemodel,our'unicorn'model.

## 3.1. Fitmodelforbestfeaturecombination

In[67]:

```
# Select features
select=SelectKBest(score_func=chi2,k=k_features_highest_score)
select.fit(X_train_poly, y_train)
X_train_poly_selected = select.transform(X_train_poly)
```

In[68]:

```
# Fit model
logreg      =       LogisticRegression(C=1)
logreg.fit(X_train_poly_selected, y_train)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
```

```
  FutureWarning)
Out[68]:

LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None,solver='warn',tol=0.0001,verbose=0,
                   warm_start=False)
In[69]:

# Model performance
scores=cross_val_score(logreg,X_train_poly_selected,y_train,cv=10)
print('CVaccuracy:%.3f+/-%.3f'%(np.mean(scores),np.std(scores)))     CV
accuracy: 0.825 +/- 0.041
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechanged  to'lbfgs'in0.22.Specifyasol   ver   to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:432:F
utureWarning:Defaultsolverwillbechangedto'lbfgs'in0.22.Specifyasol    ver    to
silence this warning.
  FutureWarning)
```

## 3.2. Learningcurve

In[70]:

```
# Plot learning curves
title="LearningCurves(LogisticRegression)" cv =
10
plot_learning_curve(logreg, title, X_train_poly_selected,
                    y_train, ylim=(0.7, 1.01), cv=cv, n_jobs=1);
```

## 3.3. Validationcurve

In[71]:

```
# Plot validation curve
title='ValidationCurve(LogisticRegression)'
param_name = 'C'
param_range = [0.001, 0.01, 0.1, 1.0, 10.0, 100.0]
cv = 10
plot_validation_curve(estimator=logreg,title=title,X=X_train_poly_selected,y=y_
train,
                      param_name=param_name,ylim=(0.5, 1.01), param_range=param_r
ange);
```

## 3.4. Submitpredictions

In[72]:

```
# Get test data set
df=pd.read_csv('../input/test.csv')
df_raw = df.copy()
```

In[73]:

```
#Transformdataset(basedonChapter2) ## 2.2
df['FamilySize']= df['SibSp'] + df['Parch']
df.drop('SibSp',axis=1,inplace=True)
df.drop('Parch',axis=1,inplace=True)
df.drop(['Name','Ticket','Cabin'],axis=1,inplace=True)

df['Title']=0
for i in df:
    df['Title']=df_raw['Name'].str.extract('([A-Za-z]+)\.',expand=False)
df_raw['Title'] = df['Title']
means=df_raw.groupby('Title')['Age'].mean() map_means
= means.to_dict()
idx_nan_age = df.loc[np.isnan(df['Age'])].index
df.loc[idx_nan_age,'Age']=df['Title'].loc[idx_nan_age].map(map_means) df['Title']
= df['Title'].map(titles_dict)
df['Title']=pd.Categorical(df['Title']) df['Imputed']

= 0
```

```
df.at[idx_nan_age.values,'Imputed'] = 1

df['Age']=pd.cut(df['Age'],bins=[0,12,50,200],labels=['Child','Adult','Elde r'])

## 2.3
passenger_id = df['PassengerId'].values
df.drop('PassengerId', axis=1, inplace=True)
df['Embarked']=pd.Categorical(df['Embarked'])
df['Pclass'] = pd.Categorical(df['Pclass'])
df= pd.get_dummies(df, drop_first=1)

df= df.fillna(df.mean())# There is one missing value in'Fare'

X=df[df.loc[:,df.columns!='Survived'].columns]

X_transformed = X.copy()
X_transformed['Fare']= boxcox(X_transformed['Fare'] + 1)[0]

scaler = MinMaxScaler()
X_transformed_scaled = scaler.fit_transform(X_transformed)

poly=PolynomialFeatures(degree=2).fit(X_transformed) X_poly
= poly.transform(X_transformed_scaled)

X_poly_selected = select.transform(X_poly)
In[74]:

# Make predictions
predictions = logreg.predict(X_poly_selected)
In[75]:#Generatesubmissionfile

submission = pd.DataFrame({ 'PassengerId': passenger_id,
                            'Survived':predictions})
submission.to_csv("submission.csv", index=False)
```

## 4. Conclusion

AsHalevyetal.(2009)noted'invariably,simplemodelsandalotofdatatrumpmore elaborate modelsbasedonlessdata.'MonicaRogati addedthat'betterdatabeatsmoredata'.Basedon these principles, the aim of this study was to improve data quality through exploratory data analysis and feature extraction. We didn't use a clever algorithm, but we explored clever techniques to make our data better.

Myexpectationisthatafterreadingthiskernel,youcanstartcompilingacookbookoftechniques in exploratory data analysis and feature extraction. These techniques will help you to obtain confidence in your data and engage with any data science problem. Also, the more you use and refine these techniques, the more you'll develop your problem solving intuition.

Now, it's your turn. Make this work yours. Select a part of this kernel and play with it. Why not trying a different feature selection process? Or what about applying a different imputation method?Thereareahundreddifferentwaysto stealthisworklikeanartist.Doit...Afterall,all unicorns started with a MVP.