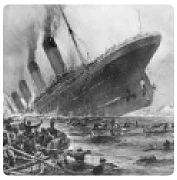


Yassine Ghouzam

Titanic Top 4% with ensemble modeling

408 voters

last run 8 months ago · Python notebook · 21615 views
using data from Titanic: Machine Learning from Disaster · Public

Tags classification feature engineering

Notebook

Titanic Top 4% with ensemble modeling

Yassine Ghouzam, PhD

13/07/2017

1. Introduction

This is my first kernel at Kaggle. I choosed the Titanic competition which is a good way to introduce feature engineering and ensemble modeling. Firstly, I will display some feature analyses then ill focus on the feature engineering. Last part concerns modeling and predicting the survival on the Titanic using an voting procedure.

This script follows three main parts:

- **Feature analysis**
- **Feature engineering**
- **Modeling**

In [1]:

```
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from collections import Counter

from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier, ExtraTreesClassifier, VotingClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV, cross_val_score, StratifiedKFold, learning_curve

sns.set(style='white', context='notebook', palette='deep')
```

2. Load and check data

2.1 Load data

In [2]:

```
# Load data
##### Load train and Test set

train = pd.read_csv("../input/train.csv")
test = pd.read_csv("../input/test.csv")
IDtest = test["PassengerId"]
```

2.2 Outlier detection

In [3]:

```
# Outlier detection

def detect_outliers(df,n,features):
    """
    Takes a dataframe df of features and returns a list of the indices
    corresponding to the observations containing more than n outliers according
    to the Tukey method.
    """
    outlier_indices = []

    # iterate over features(columns)
    for col in features:
        # 1st quartile (25%)
        Q1 = np.percentile(df[col], 25)
        # 3rd quartile (75%)
        Q3 = np.percentile(df[col],75)
        # Interquartile range (IQR)
        IQR = Q3 - Q1
```

```
        # outlier step
        outlier_step = 1.5 * IQR

        # Determine a list of indices of outliers for feature col
        outlier_list_col = df[(df[col] < Q1 - outlier_step) | (df[col] > Q3 + outlier_step )].index

        # append the found outlier indices for col to the list of outlier indices
        outlier_indices.extend(outlier_list_col)

    # select observations containing more than 2 outliers
    outlier_indices = Counter(outlier_indices)
    multiple_outliers = list( k for k, v in outlier_indices.items() if v > n )

    return multiple_outliers

# detect outliers from Age, SibSp , Parch and Fare
Outliers_to_drop = detect_outliers(train,2,["Age","SibSp","Parch","Fare"])
```

```
/opt/conda/lib/python3.6/site-packages/numpy/lib/function_base.py:4269: RuntimeWarning: Invalid value en
countered in percentile
  interpolation=interpolation)
```

Since outliers can have a dramatic effect on the prediction (espacially for regression problems), i choosed to manage them.

I used the Tukey method (Tukey JW., 1977) to detect ouliers which defines an interquartile range comprised between the 1st and 3rd quartile of the distribution values (IQR). An outlier is a row that have a feature value outside the (IQR +- an outlier step).

I decided to detect outliers from the numerical values features (Age, SibSp, Sarch and Fare). Then, i considered outliers as rows that have at least two outlied numerical values.

In [4]:

```
train.loc[Outliers_to_drop] # Show the outliers rows
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950	263.00	C23 C25 C27	S
88	89	1	1	Fortune, Miss. Mabel Helen	female	23.0	3	2	19950	263.00	C23 C25 C27	S
159	160	0	3	Sage, Master. Thomas Henry	male	NaN	8	2	CA. 2343	69.55	NaN	S
180	181	0	3	Sage, Miss. Constance Gladys	female	NaN	8	2	CA. 2343	69.55	NaN	S
201	202	0	3	Sage, Mr.	male	NaN	8	2	CA.	69.55	NaN	S

	202	0	3	Frederick	male	NaN	8	2	2343	69.55	NaN	S
324	325	0	3	Sage, Mr. George John Jr	male	NaN	8	2	CA. 2343	69.55	NaN	S
341	342	1	1	Fortune, Miss. Alice Elizabeth	female	24.0	3	2	19950	263.00	C23 C25 C27	S
792	793	0	3	Sage, Miss. Stella Anna	female	NaN	8	2	CA. 2343	69.55	NaN	S
846	847	0	3	Sage, Mr. Douglas Bullen	male	NaN	8	2	CA. 2343	69.55	NaN	S
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8	2	CA. 2343	69.55	NaN	S

We detect 10 outliers. The 28, 89 and 342 passenger have an high Ticket Fare

The 7 others have very high values of SibSP.

In [5]:

```
# Drop outliers
train = train.drop(Outliers_to_drop, axis = 0).reset_index(drop=True)
```

2.3 joining train and test set

In [6]:

```
## Join train and test datasets in order to obtain the same number of features during categorical conver
sion
train_len = len(train)
dataset =  pd.concat(objs=[train, test], axis=0).reset_index(drop=True)
```

I join train and test datasets to obtain the same number of features during categorical conversion (See feature engineering).

2.4 check for null and missing values

In [7]:

```
# Fill empty and NaNs values with NaN
dataset = dataset.fillna(np.nan)

# Check for Null values
dataset.isnull().sum()
```

Out[7]:

```
Age             256
Cabin          1007
Embarked          2
Fare              1
Name              0
Parch             0
PassengerId       0
Pclass            0
Sex               0
SibSp             0
Survived        418
Ticket            0
dtype: int64
```

Age and Cabin features have an important part of missing values.

Survived missing values correspond to the join testing dataset (Survived column doesn't exist in test set and has been replace by NaN values when concatenating the train and test set)

In [8]:

```
# Infos
train.info()
train.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 881 entries, 0 to 880
Data columns (total 12 columns):
PassengerId    881 non-null int64
Survived       881 non-null int64
Pclass         881 non-null int64
Name           881 non-null object
Sex            881 non-null object
Age            711 non-null float64
SibSp          881 non-null int64
Parch          881 non-null int64
Ticket         881 non-null object
Fare           881 non-null float64
Cabin          201 non-null object
Embarked       879 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 82.7+ KB
```

Out[8]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            170
SibSp            0
Parch            0
Ticket           0
```

```
Fare             0
Cabin          680
Embarked         2
dtype: int64
```

In [9]:

```
train.head()
```

Out[9]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embark
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

In [10]:

```
train.dtypes
```

Out[10]:

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

In [11]:

```
### Summarize data
# Summarie and statistics
```

```
# Summarie and statistics
train.describe()
```

Out[11]:

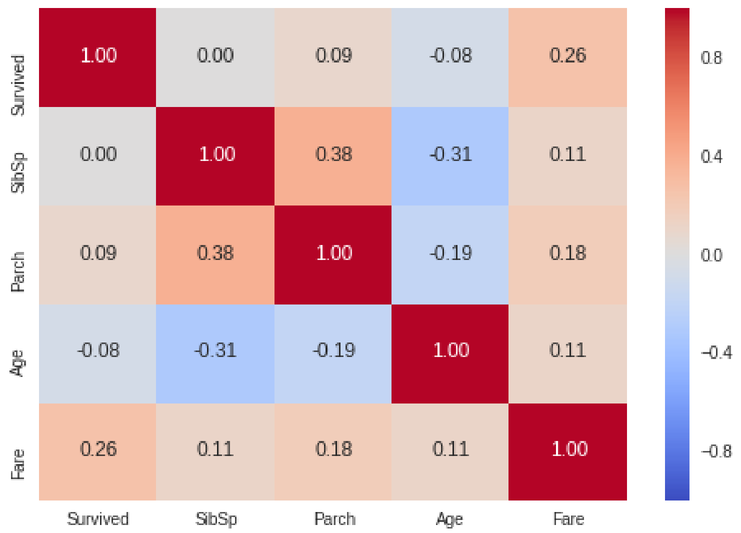
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	881.000000	881.000000	881.000000	711.000000	881.000000	881.000000	881.000000
mean	446.713961	0.385925	2.307605	29.731603	0.455165	0.363224	31.121566
std	256.617021	0.487090	0.835055	14.547835	0.871571	0.791839	47.996249
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	226.000000	0.000000	2.000000	20.250000	0.000000	0.000000	7.895800
50%	448.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.000000	1.000000	3.000000	38.000000	1.000000	0.000000	30.500000
max	891.000000	1.000000	3.000000	80.000000	5.000000	6.000000	512.329200

3. Feature analysis

3.1 Numerical values

In [12]:

```
# Correlation matrix between numerical values (SibSp Parch Age and Fare values) and Survived
g = sns.heatmap(train[["Survived","SibSp","Parch","Age","Fare"]].corr(),annot=True, fmt = ".2f", cmap =
"coolwarm")
```



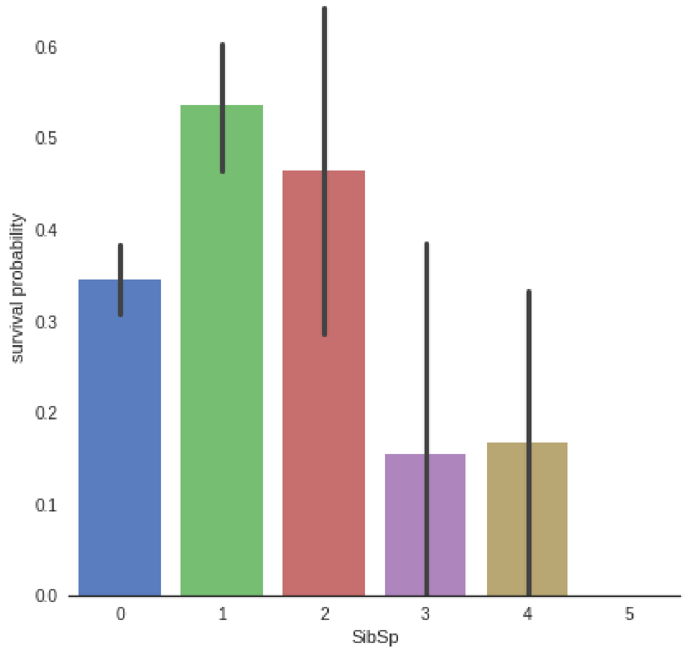
Only Fare feature seems to have a significative correlation with the survival probability.

It doesn't mean that the other features are not usefull. Subpopulations in these features can be correlated with the survival. To determine this, we need to explore in detail these features

SibSP

In [13]:

```
# Explore SibSp feature vs Survived
g = sns.factorplot(x="SibSp",y="Survived",data=train,kind="bar", size = 6 ,
palette = "muted")
g.despine(left=True)
g = g.set_ylabels("survival probability")
```



It seems that passengers having a lot of siblings/spouses have less chance to survive

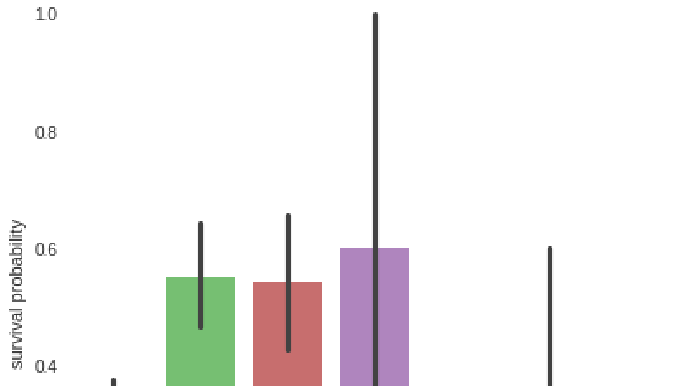
Single passengers (0 SibSP) or with two other persons (SibSP 1 or 2) have more chance to survive

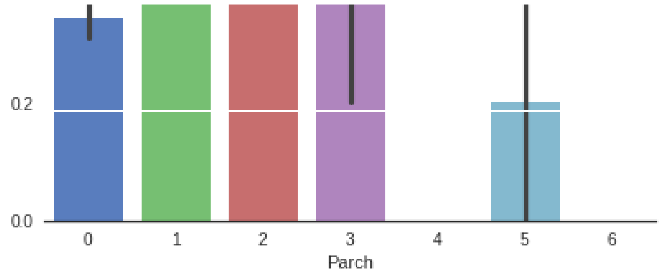
This observation is quite interesting, we can consider a new feature describing these categories (See feature engineering)

Parch

In [14]:

```
# Explore Parch feature vs Survived
g  = sns.factorplot(x="Parch",y="Survived",data=train,kind="bar", size = 6 ,
palette = "muted")
g.despine(left=True)
g = g.set_ylabels("survival probability")
```





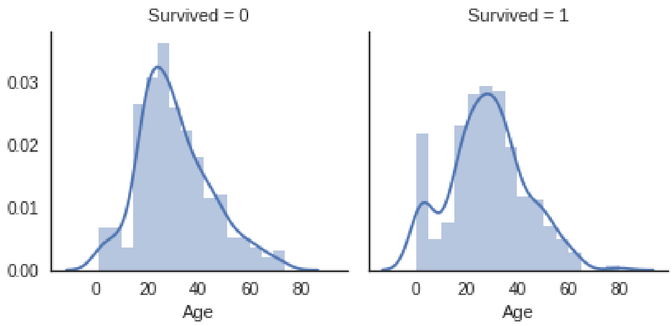
Small families have more chance to survive, more than single (Parch 0), medium (Parch 3,4) and large families (Parch 5,6).

Be carefull there is an important standard deviation in the survival of passengers with 3 parents/children

Age

In [15]:

```
# Explore Age vs Survived
g = sns.FacetGrid(train, col='Survived')
g = g.map(sns.distplot, "Age")
```



Age distribution seems to be a tailed distribution, maybe a gaussian distribution.

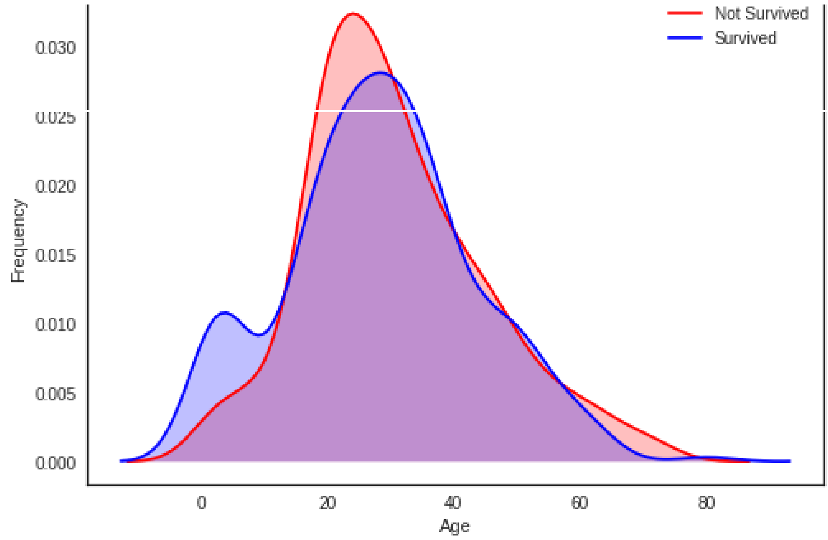
We notice that age distributions are not the same in the survived and not survived subpopulations. Indeed, there is a peak corresponding to young passengers, that have survived. We also see that passengers between 60-80 have less survived.

So, even if "Age" is not correlated with "Survived", we can see that there is age categories of passengers that of have more or less chance to survive.

It seems that very young passengers have more chance to survive.

In [16]:

```
# Explore Age distibution
g = sns.kdeplot(train["Age"][(train["Survived"] == 0) & (train["Age"].notnull())], color="Red", shade = True)
g = sns.kdeplot(train["Age"][(train["Survived"] == 1) & (train["Age"].notnull())], ax =g, color="Blue", shade= True)
g.set_xlabel("Age")
g.set_ylabel("Frequency")
g = g.legend(["Not Survived","Survived"])
```



When we superimpose the two densities , we cleary see a peak correponsing (between 0 and 5) to babies and very young childrens.

Fare

In [17]:

```
dataset["Fare"].isnull().sum()
```

Out[17]:

```
1
```

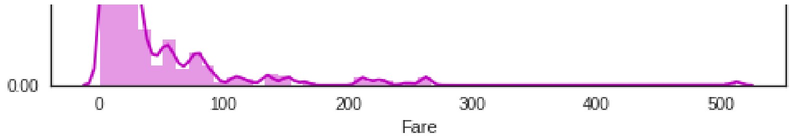
In [18]:

```
#Fill Fare missing values with the median value
dataset["Fare"] = dataset["Fare"].fillna(dataset["Fare"].median())
```

Since we have one missing value , i decided to fill it with the median value which will not have an important effect on the prediction.

In [19]:

```
# Explore Fare distribution
g = sns.distplot(dataset["Fare"], color="m", label="Skewness : %.2f"%(dataset["Fare"].skew()))
g = g.legend(loc="best")
```



As we can see, Fare distribution is very skewed. This can lead to overweigth very high values in the model, even if it is scaled.

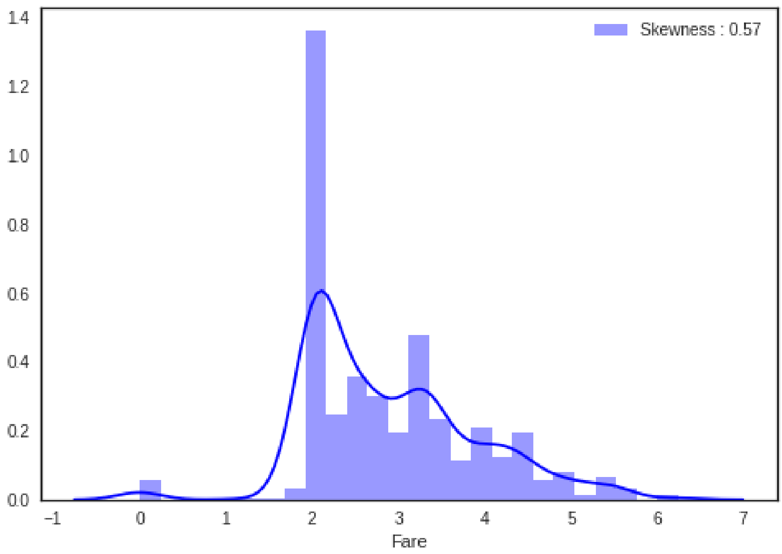
In this case, it is better to transform it with the log function to reduce this skew.

In [20]:

```
# Apply log to Fare to reduce skewness distribution
dataset["Fare"] = dataset["Fare"].map(lambda i: np.log(i) if i > 0 else 0)
```

In [21]:

```
g = sns.distplot(dataset["Fare"], color="b", label="Skewness : %.2f"%(dataset["Fare"].skew()))
g = g.legend(loc="best")
```



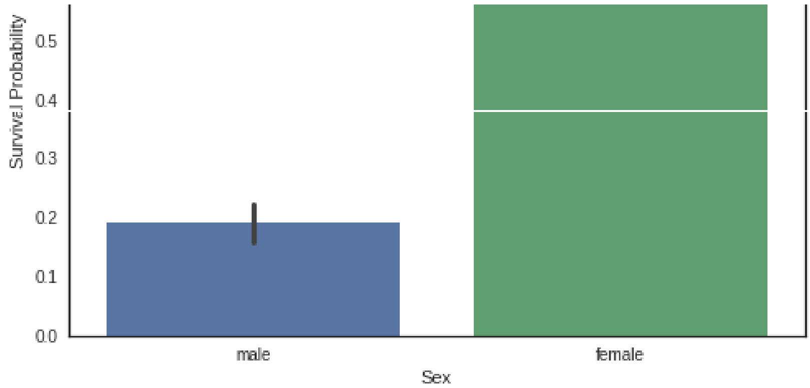
Skewness is clearly reduced after the log transformation

3.2 Categorical values

Sex

In [22]:

```
g = sns.barplot(x="Sex",y="Survived",data=train)
g = g.set_ylabel("Survival Probability")
```



In [23]:

```
train[["Sex","Survived"]].groupby('Sex').mean()
```

Out[23]:

	Survived
Sex	
female	0.747573
male	0.190559

It is clearly obvious that Male have less chance to survive than Female.

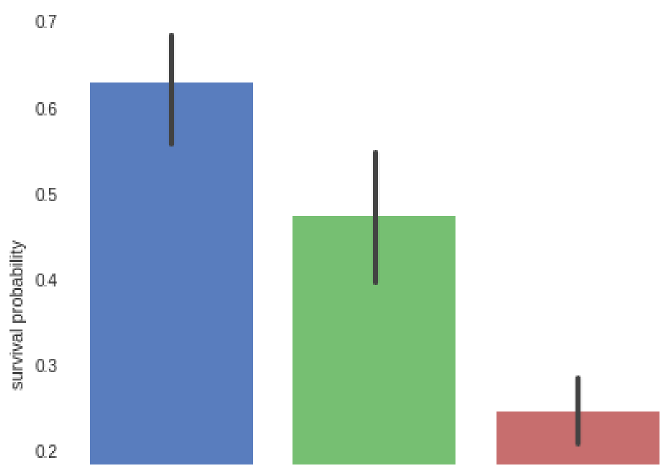
So Sex, might play an important role in the prediction of the survival.

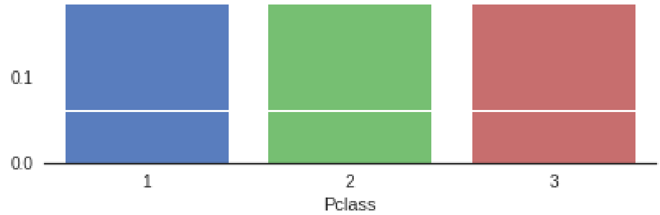
For those who have seen the Titanic movie (1997), I am sure, we all remember this sentence during the evacuation : "Women and children first".

Pclass

In [24]:

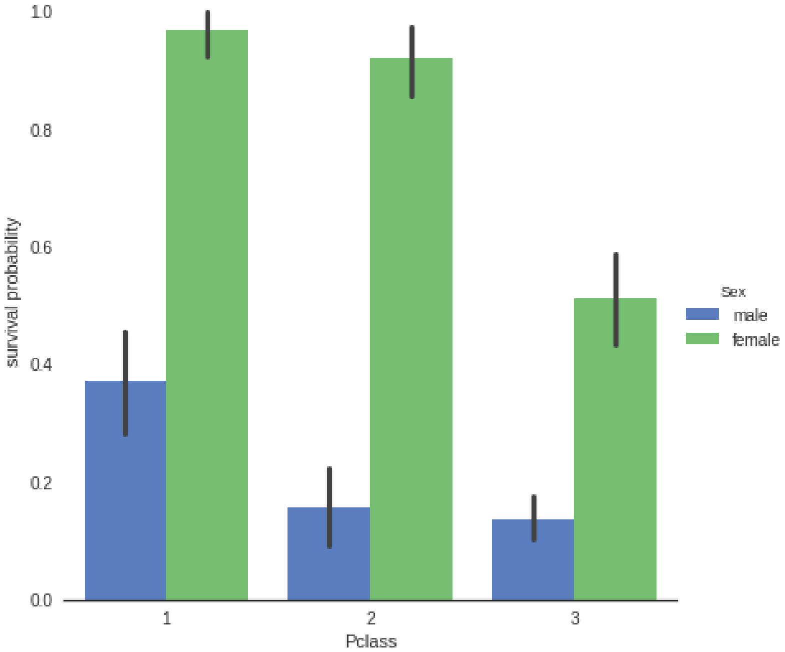
```
# Explore Pclass vs Survived
g = sns.factorplot(x="Pclass",y="Survived",data=train,kind="bar", size = 6 ,
palette = "muted")
g.despine(left=True)
g = g.set_ylabels("survival probability")
```





In [25]:

```
# Explore Pclass vs Survived by Sex
g = sns.factorplot(x="Pclass", y="Survived", hue="Sex", data=train,
                   size=6, kind="bar", palette="muted")
g.despine(left=True)
g = g.set_ylabels("survival probability")
```



The passenger survival is not the same in the 3 classes. First class passengers have more chance to survive than second class and third class passengers.

This trend is conserved when we look at both male and female passengers.

Embarked

In [26]:

```
dataset["Embarked"].isnull().sum()
```

Out[26]:

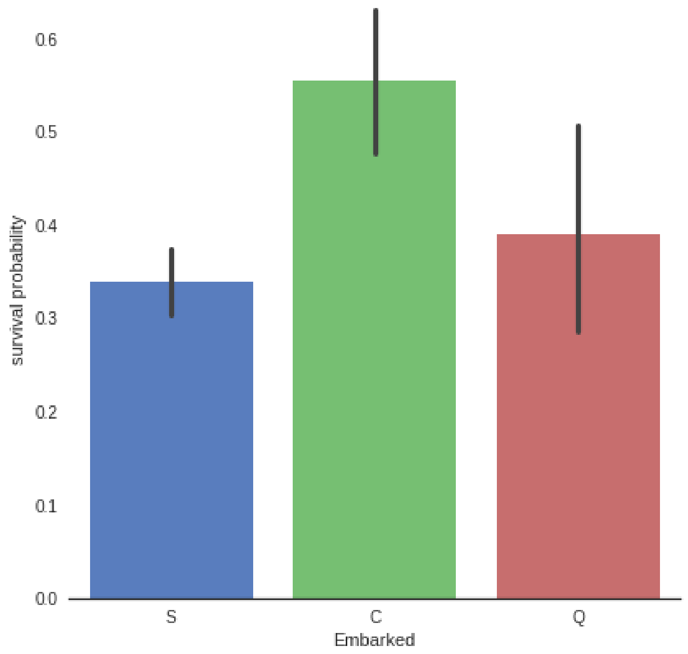
```
2
```

In [27]:

```
#Fill Embarked nan values of dataset set with 'S' most frequent value
dataset["Embarked"] = dataset["Embarked"].fillna("S")
```

Since we have two missing values , i decided to fill them with the most fequent value of "Embarked" (S).

In [28]:
```
# Explore Embarked vs Survived
g = sns.factorplot(x="Embarked", y="Survived",  data=train,
                   size=6, kind="bar", palette="muted")
g.despine(left=True)
g = g.set_ylabels("survival probability")
```

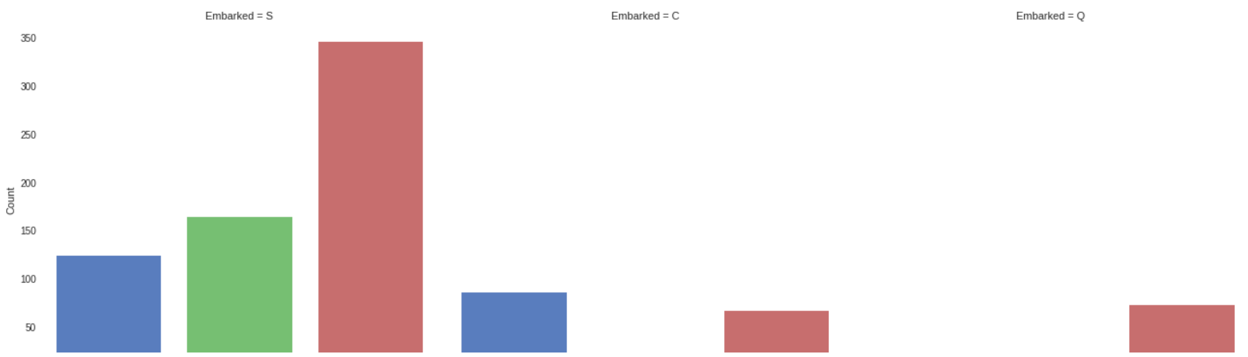


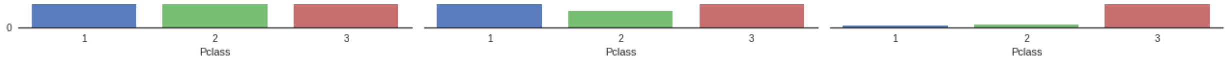
It seems that passenger coming from Cherbourg (C) have more chance to survive.

My hypothesis is that the proportion of first class passengers is higher for those who came from Cherbourg than Queenstown (Q), Southampton (S).

Let's see the Pclass distribution vs Embarked

In [29]:
```
# Explore Pclass vs Embarked
g = sns.factorplot("Pclass", col="Embarked",  data=train,
                   size=6, kind="count", palette="muted")
g.despine(left=True)
g = g.set_ylabels("Count")
```





Indeed, the third class is the most frequent for passenger coming from Southampton (S) and Queenstown (Q), whereas Cherbourg passengers are mostly in first class which have the highest survival rate.

At this point, i can't explain why first class has an higher survival rate. My hypothesis is that first class passengers were prioritised during the evacuation due to their influence.

4. Filling missing Values

4.1 Age

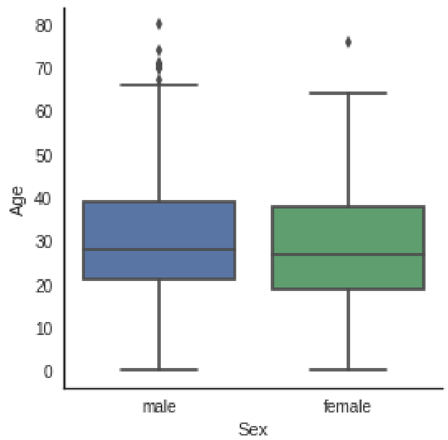
As we see, Age column contains 256 missing values in the whole dataset.

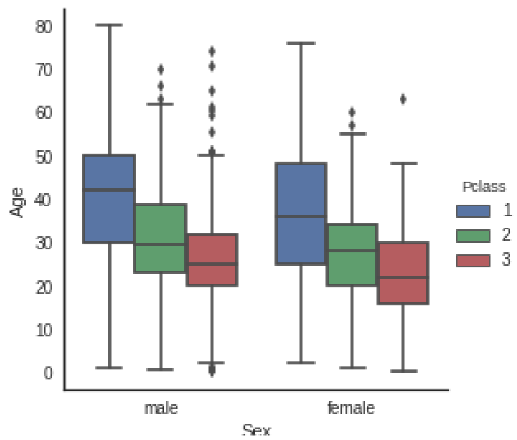
Since there is subpopulations that have more chance to survive (children for example), it is preferable to keep the age feature and to impute the missing values.

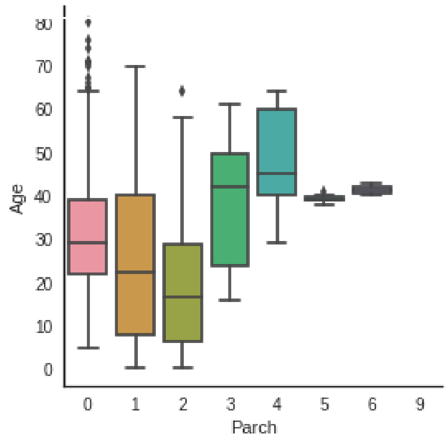
To adress this problem, i looked at the most correlated features with Age (Sex, Parch , Pclass and SibSP).

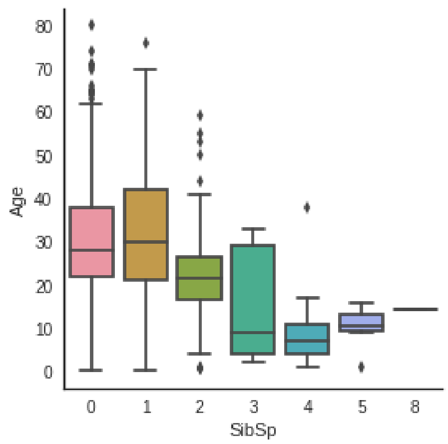
In [30]:

```
# Explore Age vs Sex, Parch , Pclass and SibSP
g = sns.factorplot(y="Age",x="Sex",data=dataset,kind="box")
g = sns.factorplot(y="Age",x="Sex",hue="Pclass", data=dataset,kind="box")
g = sns.factorplot(y="Age",x="Parch", data=dataset,kind="box")
g = sns.factorplot(y="Age",x="SibSp", data=dataset,kind="box")
```









Age distribution seems to be the same in Male and Female subpopulations, so Sex is not informative to predict Age.

However, 1rst class passengers are older than 2nd class passengers who are also older than 3rd class passengers.

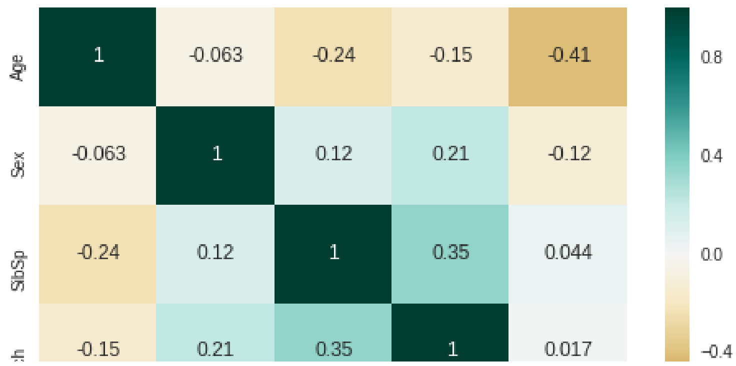
Moreover, the more a passenger has parents/children the older he is and the more a passenger has siblings/spouses the younger he is.

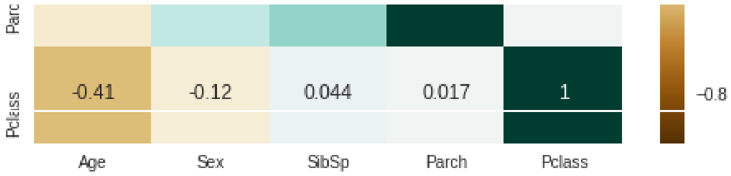
In [31]:

```
# convert Sex into categorical value 0 for male and 1 for female
dataset["Sex"] = dataset["Sex"].map({"male": 0, "female":1})
```

In [32]:

```
g = sns.heatmap(dataset[["Age","Sex","SibSp","Parch","Pclass"]].corr(),cmap="BrBG",annot=True)
```





The correlation map confirms the factorplots observations except for Parch. Age is not correlated with Sex, but is negatively correlated with Pclass, Parch and SibSp.

In the plot of Age in function of Parch, Age is growing with the number of parents / children. But the general correlation is negative.

So, i decided to use SibSP, Parch and Pclass in order to impute the missing ages.

The strategy is to fill Age with the median age of similar rows according to Pclass, Parch and SibSp.

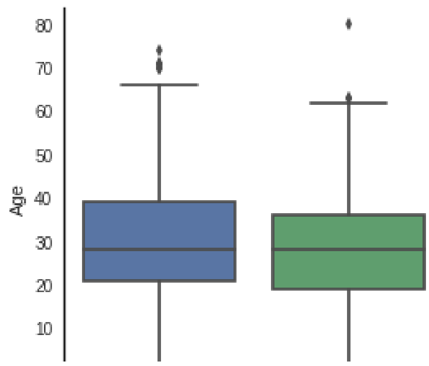
In [33]:

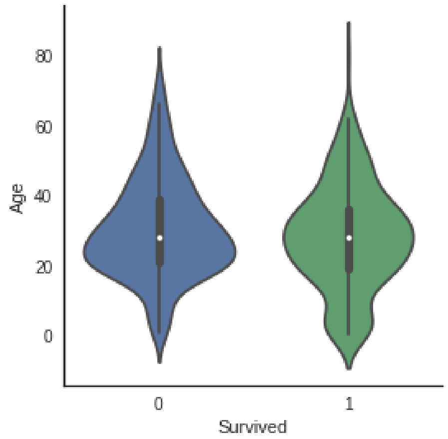
```
# Filling missing value of Age

## Fill Age with the median age of similar rows according to Pclass, Parch and SibSp
# Index of NaN age rows
index_NaN_age = list(dataset["Age"][dataset["Age"].isnull()].index)

for i in index_NaN_age :
    age_med = dataset["Age"].median()
    age_pred = dataset["Age"][((dataset['SibSp'] == dataset.iloc[i]["SibSp"]) & (dataset['Parch'] == dat
aset.iloc[i]["Parch"]) & (dataset['Pclass'] == dataset.iloc[i]["Pclass"]))].median()
    if not np.isnan(age_pred) :
        dataset['Age'].iloc[i] = age_pred
    else :
        dataset['Age'].iloc[i] = age_med
```

```
/opt/conda/lib/python3.6/site-packages/pandas/core/indexing.py:179: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing
-view-versus-copy
  self._setitem_with_indexer(indexer, value)
```

In [34]:

```
g = sns.factorplot(x="Survived", y = "Age",data = train, kind="box")
g = sns.factorplot(x="Survived", y = "Age",data = train, kind="violin")
```





No difference between median value of age in survived and not survived subpopulation.

But in the violin plot of survived passengers, we still notice that very young passengers have higher survival rate.

5. Feature engineering

5.1 Name/Title

In [35]:

```
dataset["Name"].head()
```

Out[35]:

```
0                              Braund, Mr. Owen Harris
1    Cumings, Mrs. John Bradley (Florence Briggs Th...
2                               Heikkinen, Miss. Laina
3         Futrelle, Mrs. Jacques Heath (Lily May Peel)
4                             Allen, Mr. William Henry
Name: Name, dtype: object
```

The Name feature contains information on passenger's title.

Since some passenger with distingused title may be preferred during the evacuation, it is interesting to add them to the model.

In [36]:

```
# Get Title from Name
dataset_title = [i.split(",")[1].split(".")[0].strip() for i in dataset["Name"]]
dataset["Title"] = pd.Series(dataset_title)
dataset["Title"].head()
```

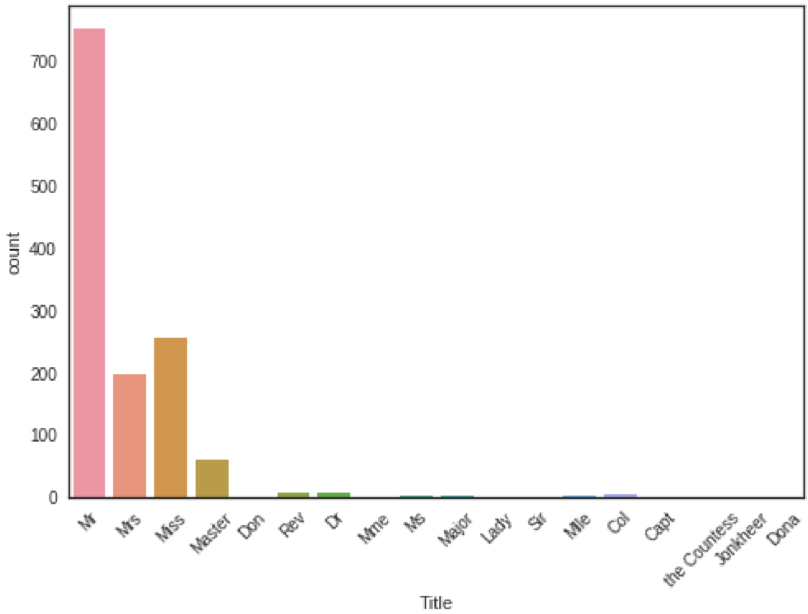
Out[36]:

```
0      Mr
1     Mrs
2    Miss
```

```
3      Mrs
4       Mr
Name: Title, dtype: object
```

In [37]:

```
g = sns.countplot(x="Title",data=dataset)
g = plt.setp(g.get_xticklabels(), rotation=45)
```



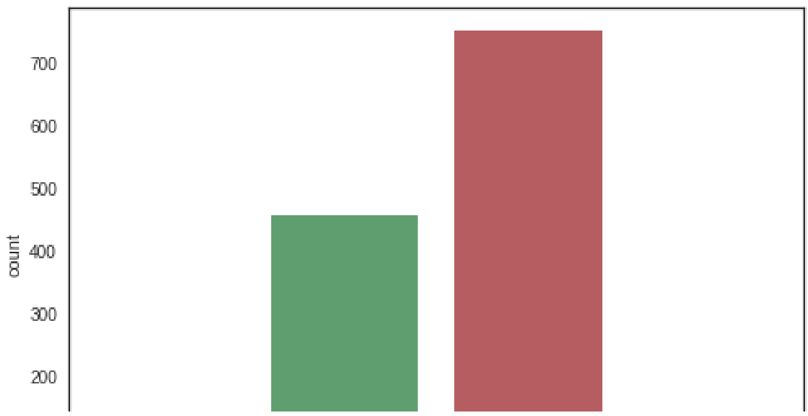
There is 17 titles in the dataset, most of them are very rare and we can group them in 4 categories.

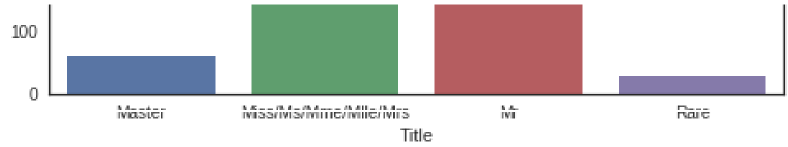
In [38]:

```
# Convert to categorical values Title
dataset["Title"] = dataset["Title"].replace(['Lady', 'the Countess','Countess','Capt', 'Col','Don', 'Dr'
, 'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')
dataset["Title"] = dataset["Title"].map({"Master":0, "Miss":1, "Ms" : 1 , "Mme":1, "Mlle":1, "Mrs":1, "M
r":2, "Rare":3})
dataset["Title"] = dataset["Title"].astype(int)
```

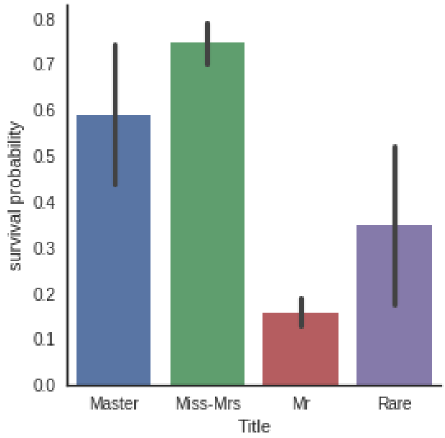
In [39]:

```
g = sns.countplot(dataset["Title"])
g = g.set_xticklabels(["Master","Miss/Ms/Mme/Mlle/Mrs","Mr","Rare"])
```





In [40]:
```
g = sns.factorplot(x="Title",y="Survived",data=dataset,kind="bar")
g = g.set_xticklabels(["Master","Miss-Mrs","Mr","Rare"])
g = g.set_ylabels("survival probability")
```



"Women and children first"

It is interesting to note that passengers with rare title have more chance to survive.

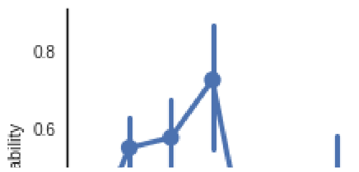
In [41]:
```
# Drop Name variable
dataset.drop(labels = ["Name"], axis = 1, inplace = True)
```

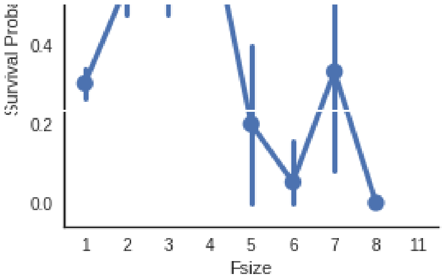
5.2 Family size

We can imagine that large families will have more difficulties to evacuate, looking for theirs sisters/brothers/parents during the evacuation. So, i choosed to create a "Fize" (family size) feature which is the sum of SibSp , Parch and 1 (including the passenger).

In [42]:
```
# Create a family size descriptor from SibSp and Parch
dataset["Fsize"] = dataset["SibSp"] + dataset["Parch"] + 1
```

In [43]:
```
g = sns.factorplot(x="Fsize",y="Survived",data = dataset)
g = g.set_ylabels("Survival Probability")
```





The family size seems to play an important role, survival probability is worst for large families.

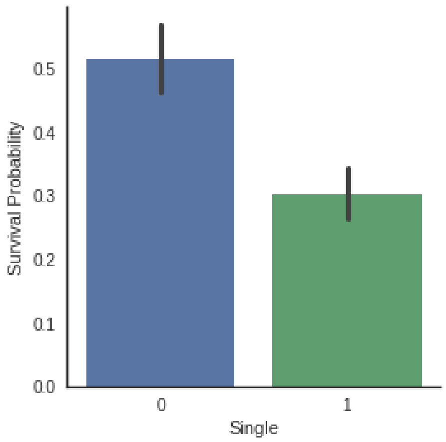
Additionally, i decided to created 4 categories of family size.

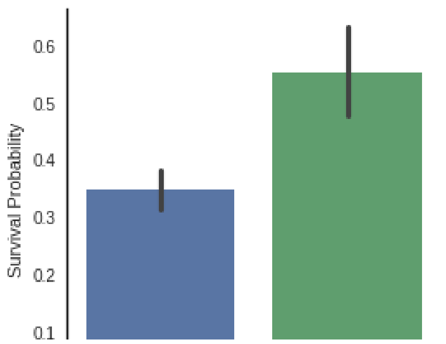
In [44]:

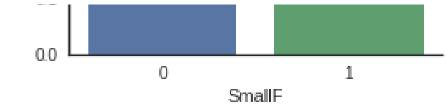
```
# Create new feature of family size
dataset['Single'] = dataset['Fsize'].map(lambda s: 1 if s == 1 else 0)
dataset['SmallF'] = dataset['Fsize'].map(lambda s: 1 if  s == 2  else 0)
dataset['MedF'] = dataset['Fsize'].map(lambda s: 1 if 3 <= s <= 4 else 0)
dataset['LargeF'] = dataset['Fsize'].map(lambda s: 1 if s >= 5 else 0)
```

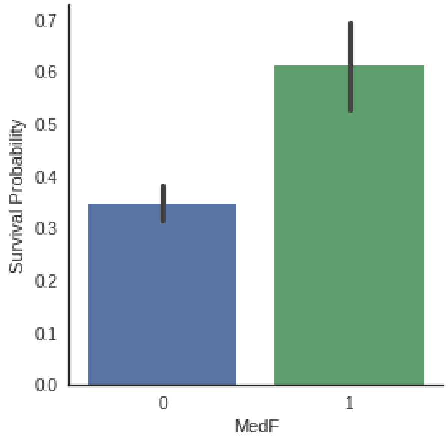
In [45]:

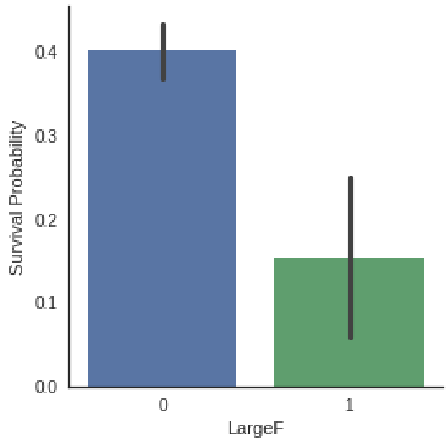
```
g = sns.factorplot(x="Single",y="Survived",data=dataset,kind="bar")
g = g.set_ylabels("Survival Probability")
g = sns.factorplot(x="SmallF",y="Survived",data=dataset,kind="bar")
g = g.set_ylabels("Survival Probability")
g = sns.factorplot(x="MedF",y="Survived",data=dataset,kind="bar")
g = g.set_ylabels("Survival Probability")
g = sns.factorplot(x="LargeF",y="Survived",data=dataset,kind="bar")
g = g.set_ylabels("Survival Probability")
```











Factorplots of family size categories show that Small and Medium families have more chance to survive than single passenger and large families.

In [46]:

```
# convert to indicator values Title and Embarked
dataset = pd.get_dummies(dataset, columns = ["Title"])
dataset = pd.get_dummies(dataset, columns = ["Embarked"], prefix="Em")
```

In [47]:

```
dataset.head()
```

Out[47]:

	Age	Cabin	Fare	Parch	PassengerId	Pclass	Sex	SibSp	Survived	Ticket	...	SmallF	MedF
0	22.0	NaN	1.981001	0	1	3	0	1	0.0	A/5 21171	...	1	0
1	38.0	C85	4.266662	0	2	1	1	1	1.0	PC 17599	...	1	0
2	26.0	NaN	2.070022	0	3	3	1	0	1.0	STON/O2. 3101282	...	0	0
3	35.0	C123	3.972177	0	4	1	1	1	1.0	113803	...	1	0
4	35.0	NaN	2.085672	0	5	3	0	0	0.0	373450	...	0	0

5 rows × 22 columns

At this stage, we have 22 features.

5.3 Cabin

In [48]:

```
dataset["Cabin"].head()
```

Out[48]:

```
0     NaN
1     C85
2     NaN
3    C123
4     NaN
Name: Cabin, dtype: object
```

In [49]:

```
dataset["Cabin"].describe()
```

Out[49]:

```
count                 292
unique                186
top       B57 B59 B63 B66
freq                    5
Name: Cabin, dtype: object
```

In [50]:

```
dataset["Cabin"].isnull().sum()
```

Out[50]:

```
1007
```

The Cabin feature column contains 292 values and 1007 missing values.

I supposed that passengers without a cabin have a missing value displayed instead of the cabin number.

In [51]:

```
dataset["Cabin"][dataset["Cabin"].notnull()].head()
```

Out[51]:

```
1      C85
3     C123
6      E46
10      G6
11    C103
Name: Cabin, dtype: object
```

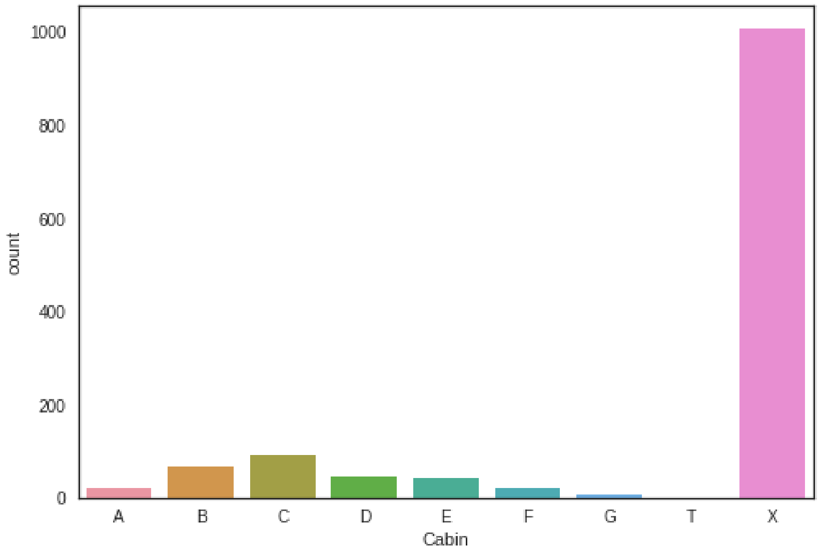
In [52]:

```
# Replace the Cabin number by the type of cabin 'X' if not
dataset["Cabin"] = pd.Series([i[0] if not pd.isnull(i) else 'X' for i in dataset['Cabin'] ])
```

The first letter of the cabin indicates the Desk, i choosed to keep this information only, since it indicates the probable location of the passenger in the Titanic.

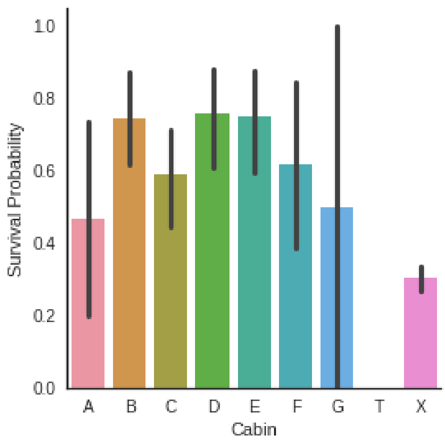
In [53]:

```
g = sns.countplot(dataset["Cabin"],order=['A','B','C','D','E','F','G','T','X'])
```



In [54]:

```
g = sns.factorplot(y="Survived",x="Cabin",data=dataset,kind="bar",order=['A','B','C','D','E','F','G','T'
,'X'])
g = g.set_ylabels("Survival Probability")
```



Because of the low number of passenger that have a cabin, survival probabilities have an important standard deviation and we can't distinguish between survival probability of passengers in the different desks.

But we can see that passengers with a cabin have generally more chance to survive than passengers without (X).

It is particularly true for cabin B, C, D, E and F.

In [55]:
```
dataset = pd.get_dummies(dataset, columns = ["Cabin"],prefix="Cabin")
```

5.4 Ticket

In [56]:
```
dataset["Ticket"].head()
```

Out[56]:
```
0           A/5 21171
1            PC 17599
2    STON/O2. 3101282
3              113803
4              373450
Name: Ticket, dtype: object
```

It could mean that tickets sharing the same prefixes could be booked for cabins placed together. It could therefore lead to the actual placement of the cabins within the ship.

Tickets with same prefixes may have a similar class and survival.

So i decided to replace the Ticket feature column by the ticket prefix. Which may be more informative.

In [57]:
```
## Treat Ticket by extracting the ticket prefix. When there is no prefix it returns X.

Ticket = []
for i in list(dataset.Ticket):
    if not i.isdigit() :
        Ticket.append(i.replace(".","").replace("/","").strip().split(' ')[0]) #Take prefix
    else:
        Ticket.append("X")

dataset["Ticket"] = Ticket
dataset["Ticket"].head()
```

Out[57]:
```
0        A5
1        PC
2    STONO2
3         X
4         X
Name: Ticket, dtype: object
```

In [58]:
```
dataset = pd.get_dummies(dataset, columns = ["Ticket"], prefix="T")
```

In [59]:
```
# Create categorical values for Pclass
dataset["Pclass"] = dataset["Pclass"].astype("category")
dataset = pd.get_dummies(dataset, columns = ["Pclass"],prefix="Pc")
```

```
dataset = pd.get_dummies(dataset, columns = ["Pclass"],prefix="Pc")
```

In [60]:

```
# Drop useless variables
dataset.drop(labels = ["PassengerId"], axis = 1, inplace = True)
```

In [61]:

```
dataset.head()
```

Out[61]:

	Age	Fare	Parch	Sex	SibSp	Survived	Fsize	Single	SmallF	MedF	...	T_STONO	T_STONO2	T_
0	22.0	1.981001	0	0	1	0.0	2	0	1	0	...	0	0	0
1	38.0	4.266662	0	1	1	1.0	2	0	1	0	...	0	0	0
2	26.0	2.070022	0	1	0	1.0	1	1	0	0	...	0	1	0
3	35.0	3.972177	0	1	1	1.0	2	0	1	0	...	0	0	0
4	35.0	2.085672	0	0	0	0.0	1	1	0	0	...	0	0	0

5 rows × 67 columns

6. MODELING

In [62]:

```
## Separate train dataset and test dataset

train = dataset[:train_len]
test = dataset[train_len:]
test.drop(labels=["Survived"],axis = 1,inplace=True)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing
-view-versus-copy
  """
```

In [63]:

```
## Separate train features and label

train["Survived"] = train["Survived"].astype(int)

Y_train = train["Survived"]

X_train = train.drop(labels = ["Survived"],axis = 1)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing
-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imports until
```

6.1 Simple modeling

6.1.1 Cross validate models

I compared 10 popular classifiers and evaluate the mean accuracy of each of them by a stratified kfold cross validation procedure.

- SVC
- Decision Tree
- AdaBoost
- Random Forest
- Extra Trees
- Gradient Boosting
- Multiple layer perceprton (neural network)
- KNN
- Logistic regression
- Linear Discriminant Analysis

In [64]:

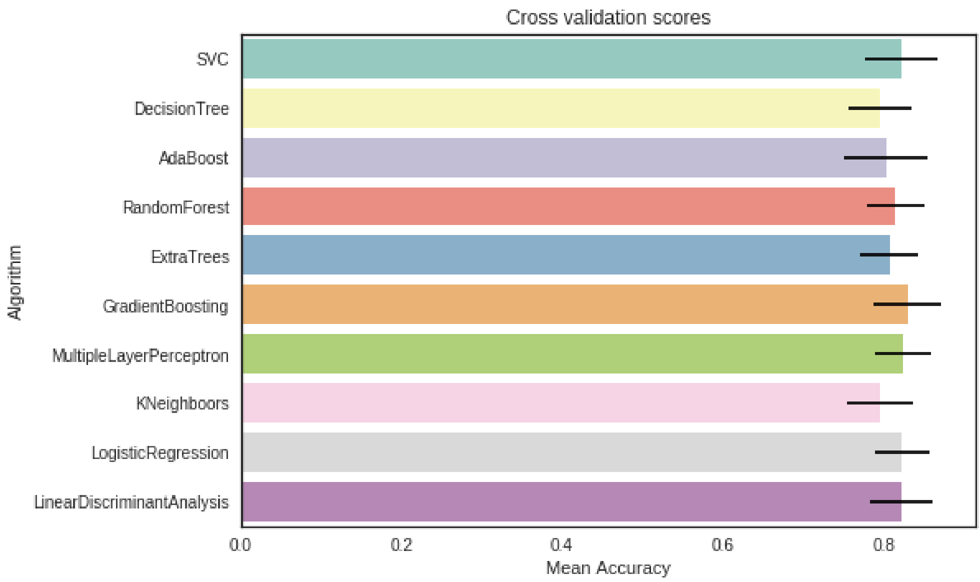
```
# Cross validate model with Kfold stratified cross val
kfold = StratifiedKFold(n_splits=10)
```

In [65]:

```
# Modeling step Test differents algorithms
random_state = 2
classifiers = []
classifiers.append(SVC(random_state=random_state))
classifiers.append(DecisionTreeClassifier(random_state=random_state))
classifiers.append(AdaBoostClassifier(DecisionTreeClassifier(random_state=random_state),random_state=ran
dom_state,learning_rate=0.1))
classifiers.append(RandomForestClassifier(random_state=random_state))
classifiers.append(ExtraTreesClassifier(random_state=random_state))
classifiers.append(GradientBoostingClassifier(random_state=random_state))
classifiers.append(MLPClassifier(random_state=random_state))
classifiers.append(KNeighborsClassifier())
classifiers.append(LogisticRegression(random_state = random_state))
classifiers.append(LinearDiscriminantAnalysis())

cv_results = []
for classifier in classifiers :
    cv_results.append(cross_val_score(classifier, X_train, y = Y_train, scoring = "accuracy", cv = kfold
, n_jobs=4))

cv_means = []
cv_std = []
for cv_result in cv_results:
    cv_means.append(cv_result.mean())
    cv_std.append(cv_result.std())
```

```
cv_res = pd.DataFrame({"CrossValMeans":cv_means,"CrossValerrors": cv_std,"Algorithm":["SVC","DecisionTre
e","AdaBoost",
"RandomForest","ExtraTrees","GradientBoosting","MultipleLayerPerceptron","KNeighboors","LogisticRegressi
on","LinearDiscriminantAnalysis"]})

g = sns.barplot("CrossValMeans","Algorithm",data = cv_res, palette="Set3",orient = "h",**{'xerr':cv_std
})
g.set_xlabel("Mean Accuracy")
g = g.set_title("Cross validation scores")
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/discriminant_analysis.py:388: UserWarning: Variables are
collinear.
  warnings.warn("Variables are collinear.")
/opt/conda/lib/python3.6/site-packages/sklearn/discriminant_analysis.py:388: UserWarning: Variables are
collinear.
  warnings.warn("Variables are collinear.")
/opt/conda/lib/python3.6/site-packages/sklearn/discriminant_analysis.py:388: UserWarning: Variables are
collinear.
  warnings.warn("Variables are collinear.")
/opt/conda/lib/python3.6/site-packages/sklearn/discriminant_analysis.py:388: UserWarning: Variables are
collinear.
  warnings.warn("Variables are collinear.")
```



I decided to choose the SVC, AdaBoost, RandomForest , ExtraTrees and the GradientBoosting classifiers for the ensemble modeling.

6.1.2 Hyperparameter tunning for best models

I performed a grid search optimization for AdaBoost, ExtraTrees , RandomForest, GradientBoosting and SVC classifiers.

I set the "n_jobs" parameter to 4 since i have 4 cpu . The computation time is clearly reduced.

But be carefull, this step can take a long time, i took me 15 min in total on 4 cpu.

In [66]:

```
### META MODELING  WITH ADABOOST, RF, EXTRATREES and GRADIENTBOOSTING

# Adaboost
DTC = DecisionTreeClassifier()

adaDTC = AdaBoostClassifier(DTC, random_state=7)

ada_param_grid = {"base_estimator__criterion" : ["gini", "entropy"],
              "base_estimator__splitter" :   ["best", "random"],
              "algorithm" : ["SAMME","SAMME.R"],
              "n_estimators" :[1,2],
              "learning_rate":  [0.0001, 0.001, 0.01, 0.1, 0.2, 0.3,1.5]}

gsadaDTC = GridSearchCV(adaDTC,param_grid = ada_param_grid, cv=kfold, scoring="accuracy", n_jobs= 4, verbose = 1)

gsadaDTC.fit(X_train,Y_train)

ada_best = gsadaDTC.best_estimator_
```

```
Fitting 10 folds for each of 112 candidates, totalling 1120 fits

[Parallel(n_jobs=4)]: Done 608 tasks      | elapsed:    4.1s
[Parallel(n_jobs=4)]: Done 1120 out of 1120 | elapsed:    7.3s finished
```

In [67]:

```
gsadaDTC.best_score_
```

Out[67]:

```
0.82406356413166859
```

In [68]:

```
#ExtraTrees
ExtC = ExtraTreesClassifier()


## Search grid for optimal parameters
ex_param_grid = {"max_depth": [None],
              "max_features": [1, 3, 10],
              "min_samples_split": [2, 3, 10],
              "min_samples_leaf": [1, 3, 10],
              "bootstrap": [False],
              "n_estimators" :[100,300],
              "criterion": ["gini"]}


gsExtC = GridSearchCV(ExtC,param_grid = ex_param_grid, cv=kfold, scoring="accuracy", n_jobs= 4, verbose = 1)

gsExtC.fit(X_train,Y_train)

ExtC_best = gsExtC.best_estimator_
```

```
# Best score
gsExtC.best_score_
```

```
Fitting 10 folds for each of 54 candidates, totalling 540 fits


[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:   10.8s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:   37.1s
[Parallel(n_jobs=4)]: Done 442 tasks      | elapsed:  1.5min
[Parallel(n_jobs=4)]: Done 540 out of 540 | elapsed:  1.8min finished
```

Out[68]:

```
0.82973893303064694
```

In [69]:

```
# RFC Parameters tunning
RFC = RandomForestClassifier()


## Search grid for optimal parameters
rf_param_grid = {"max_depth": [None],
              "max_features": [1, 3, 10],
              "min_samples_split": [2, 3, 10],
              "min_samples_leaf": [1, 3, 10],
              "bootstrap": [False],
              "n_estimators" :[100,300],
              "criterion": ["gini"]}


gsRFC = GridSearchCV(RFC,param_grid = rf_param_grid, cv=kfold, scoring="accuracy", n_jobs= 4, verbose = 1)

gsRFC.fit(X_train,Y_train)

RFC_best = gsRFC.best_estimator_

# Best score
gsRFC.best_score_
```

```
Fitting 10 folds for each of 54 candidates, totalling 540 fits


[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:   11.5s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:   39.3s
[Parallel(n_jobs=4)]: Done 442 tasks      | elapsed:  1.6min
[Parallel(n_jobs=4)]: Done 540 out of 540 | elapsed:  2.0min finished
```

Out[69]:

```
0.83427922814982969
```

In [70]:

```
# Gradient boosting tunning

GBC = GradientBoostingClassifier()
gb_param_grid = {'loss' : ["deviance"],
              'n_estimators' : [100,200,300],
```

```
              'n_estimators' : [100,200,300],
              'learning_rate': [0.1, 0.05, 0.01],
              'max_depth': [4, 8],
              'min_samples_leaf': [100,150],
              'max_features': [0.3, 0.1]
              }

gsGBC = GridSearchCV(GBC,param_grid = gb_param_grid, cv=kfold, scoring="accuracy", n_jobs= 4, verbose =
1)

gsGBC.fit(X_train,Y_train)

GBC_best = gsGBC.best_estimator_

# Best score
gsGBC.best_score_
```

```
Fitting 10 folds for each of 72 candidates, totalling 720 fits


[Parallel(n_jobs=4)]: Done  76 tasks      | elapsed:    6.3s
[Parallel(n_jobs=4)]: Done 376 tasks      | elapsed:   29.6s
[Parallel(n_jobs=4)]: Done 720 out of 720 | elapsed:   56.6s finished
```

Out[70]:

```
0.83087400681044266
```

In [71]:

```
### SVC classifier
SVMC = SVC(probability=True)
svc_param_grid = {'kernel': ['rbf'],
                  'gamma': [ 0.001, 0.01, 0.1, 1],
                  'C': [1, 10, 50, 100,200,300, 1000]}

gsSVMC = GridSearchCV(SVMC,param_grid = svc_param_grid, cv=kfold, scoring="accuracy", n_jobs= 4, verbose
 = 1)

gsSVMC.fit(X_train,Y_train)

SVMC_best = gsSVMC.best_estimator_

# Best score
gsSVMC.best_score_
```

```
Fitting 10 folds for each of 28 candidates, totalling 280 fits


[Parallel(n_jobs=4)]: Done  42 tasks      | elapsed:    8.3s
[Parallel(n_jobs=4)]: Done 192 tasks      | elapsed:   38.6s
[Parallel(n_jobs=4)]: Done 280 out of 280 | elapsed:  1.0min finished
```

Out[71]:

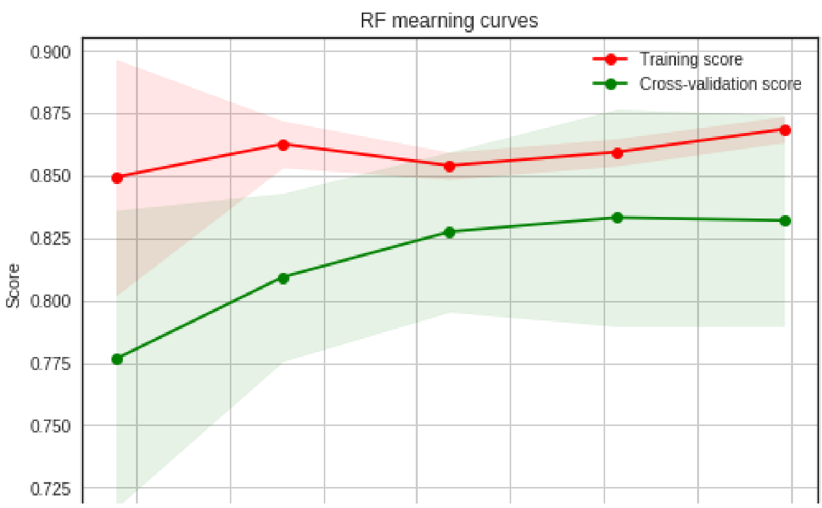
```
0.83314415437003408
```

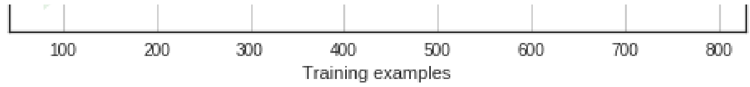
6.1.3 Plot learning curves

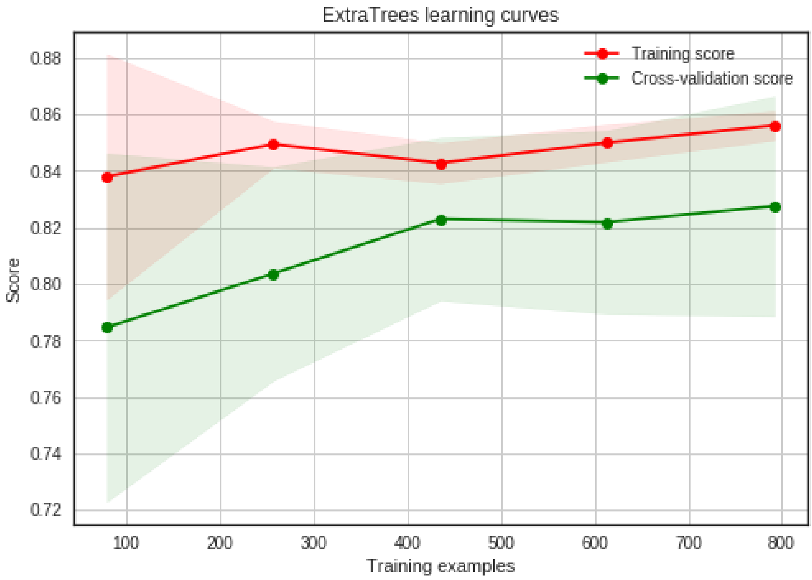
Learning curves are a good way to see the overfitting effect on the training set and the effect of the training size on the accuracy.

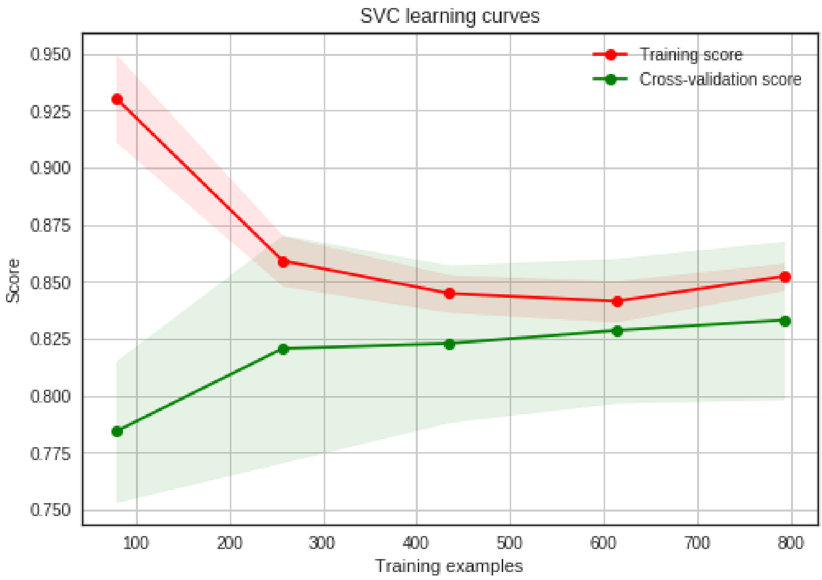
In [72]:

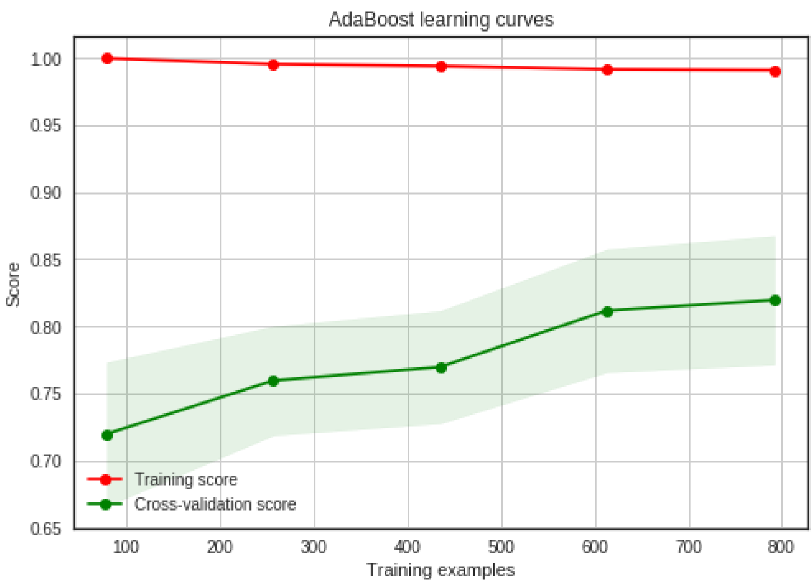
```
def plot_learning_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=-1, train_sizes=np.linspace(.1, 1.0, 5)):
    """Generate a simple plot of the test and training learning curve"""
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    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="Cross-validation score")

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

g = plot_learning_curve(gsRFC.best_estimator_,"RF mearning curves",X_train,Y_train,cv=kfold)
g = plot_learning_curve(gsExtC.best_estimator_,"ExtraTrees learning curves",X_train,Y_train,cv=kfold)
g = plot_learning_curve(gsSVMC.best_estimator_,"SVC learning curves",X_train,Y_train,cv=kfold)
g = plot_learning_curve(gsadaDTC.best_estimator_,"AdaBoost learning curves",X_train,Y_train,cv=kfold)
g = plot_learning_curve(gsGBC.best_estimator_,"GradientBoosting learning curves",X_train,Y_train,cv=kfol
d)
```

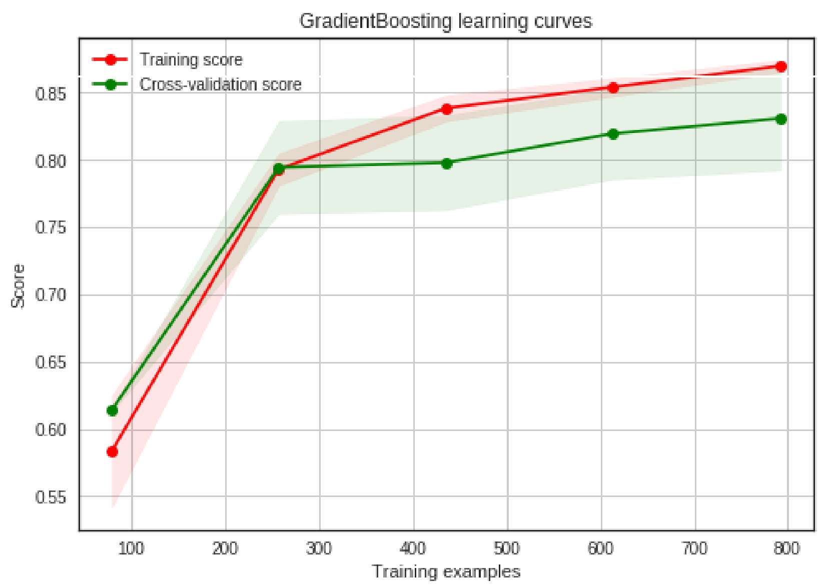



100
200
300
400
500
600
700
800
Training examples


ExtraTrees learning curves
Training score
Cross-validation score
Score
0.88
0.86
0.84
0.82
0.80
0.78
0.76
0.74
0.72
100
200
300
400
500
600
700
800
Training examples


SVC learning curves
Training score
Cross-validation score
Score
0.950
0.925
0.900
0.875
0.850
0.825
0.800
0.775
0.750
100
200
300
400
500
600
700
800
Training examples


AdaBoost learning curves
Training score
Cross-validation score
Score
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
100
200
300
400
500
600
700
800
Training examples



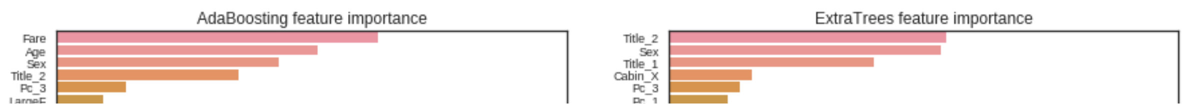
SVC and ExtraTrees classifiers seem to better generalize the prediction since the training and cross-validation curves are close together.

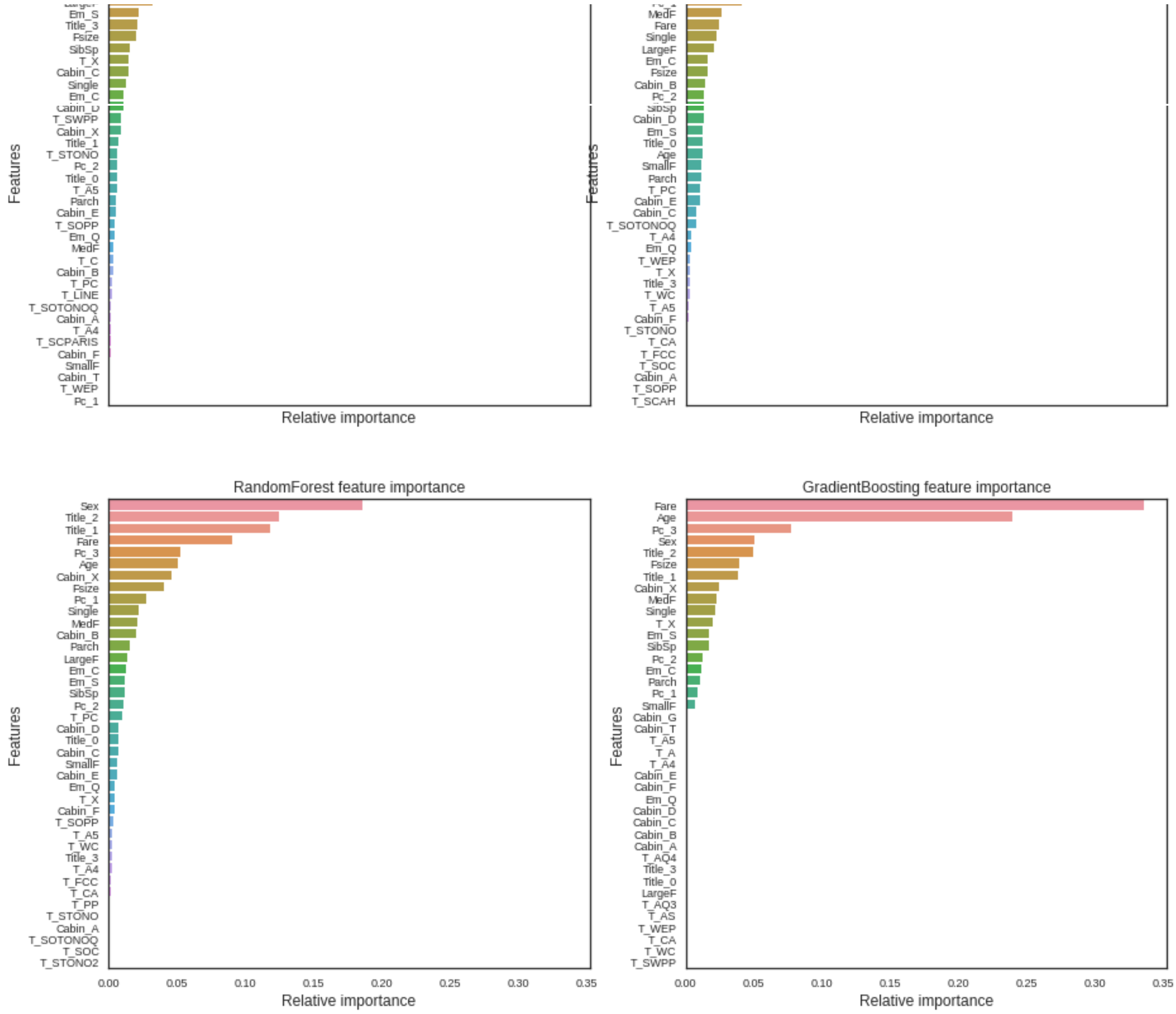
6.1.4 Feature importance of tree based classifiers

In order to see the most informative features for the prediction of passengers survival, i displayed the feature importance for the 4 tree based classifiers.

In [73]:

```
nrows = ncols = 2
fig, axes = plt.subplots(nrows = nrows, ncols = ncols, sharex="all", figsize=(15,15))

names_classifiers = [("AdaBoosting", ada_best),("ExtraTrees",ExtC_best),("RandomForest",RFC_best),("Grad
ientBoosting",GBC_best)]

nclassifier = 0
for row in range(nrows):
    for col in range(ncols):
        name = names_classifiers[nclassifier][0]
        classifier = names_classifiers[nclassifier][1]
        indices = np.argsort(classifier.feature_importances_)[::-1][:40]
        g = sns.barplot(y=X_train.columns[indices][:40],x = classifier.feature_importances_[indices][:40
] , orient='h',ax=axes[row][col])
        g.set_xlabel("Relative importance",fontsize=12)
        g.set_ylabel("Features",fontsize=12)
        g.tick_params(labelsize=9)
        g.set_title(name + " feature importance")
        nclassifier += 1
```





I plot the feature importance for the 4 tree based classifiers (Adaboost, ExtraTrees, RandomForest and GradientBoosting).

We note that the four classifiers have different top features according to the relative importance. It means that their predictions are not based on the same features. Nevertheless, they share some common important features for the classification , for example 'Fare', 'Title_2', 'Age' and 'Sex'.

Title_2 which indicates the Mrs/Mlle/Mme/Miss/Ms category is highly correlated with Sex.

We can say that:

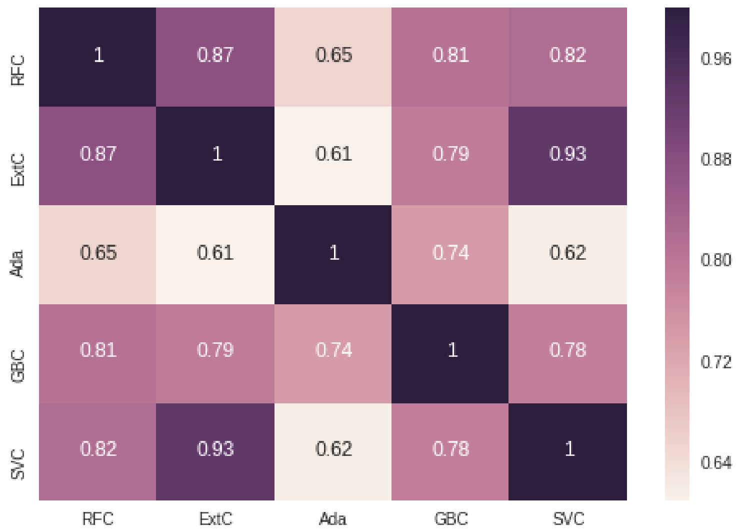
- Pc_1, Pc_2, Pc_3 and Fare refer to the general social standing of passengers.
- Sex and Title_2 (Mrs/Mlle/Mme/Miss/Ms) and Title_3 (Mr) refer to the gender.
- Age and Title_1 (Master) refer to the age of passengers.
- Fsize, LargeF, MedF, Single refer to the size of the passenger family.

According to the feature importance of this 4 classifiers, the prediction of the survival seems to be more associated with the Age, the Sex, the family size and the social standing of the passengers more than the location in the boat.

In [74]:

```
test_Survived_RFC = pd.Series(RFC_best.predict(test), name="RFC")
test_Survived_ExtC = pd.Series(ExtC_best.predict(test), name="ExtC")
test_Survived_SVMC = pd.Series(SVMC_best.predict(test), name="SVC")
test_Survived_AdaC = pd.Series(ada_best.predict(test), name="Ada")
test_Survived_GBC = pd.Series(GBC_best.predict(test), name="GBC")
```

```
test_Survived_GBC = pd.Series(GBC_best.predict(test), name="GBC")


# Concatenate all classifier results
ensemble_results = pd.concat([test_Survived_RFC,test_Survived_ExtC,test_Survived_AdaC,test_Survived_GBC,
 test_Survived_SVMC],axis=1)


g= sns.heatmap(ensemble_results.corr(),annot=True)
```



The prediction seems to be quite similar for the 5 classifiers except when Adaboost is compared to the others classifiers.

The 5 classifiers give more or less the same prediction but there is some differences. Theses differences between the 5 classifier predictions are sufficient to consider an ensembling vote.

Did you find this Kernel useful?
Show your appreciation with an upvote

408

Comments (100)   All Comments   Sort by Hotness

Please sign in to leave a comment.

ClovisFilho • Posted on Latest Version • a month ago • Options   6

This is, by far, the best text about the subject I've ever seen so far. I've learned a lot from it! Thank you, sir.

yuquan.li • Posted on Latest Version • 3 months ago • Options   0

You write a excellent kernel for us.Thank you. But I have a little problem with your kernel, there are some abnormal value in the raw dataset. Why not try random forest regression?

Nilzone • Posted on Latest Version • 6 months ago • Options

1

Great Kernel!! I learned a lot. I only have one question; The final dataset you used for the modelling-part, contains mostly 0's and 1's, except the Age and Fare-features. Would it have any impact to scale it with MinMaxScaler for example? Or is it enough to log-transform only the skewed features (like Fare in the case)?

Fotis Savva • Posted on Latest Version • 7 months ago • Options

4

Hello, great kernel and great feature engineering work. Am I right in assuming that features that were separated into Indicator Variables is to avoid the assumption of a distance between One-Hot Encoded variables ?

Chih-Ming Wang • Posted on Latest Version • 7 months ago • Options

1

Thanks for sharing. In 6.1.3, you indicated that both Adaboost and GradientBoost tend to overfit. The reason for Adaboost is obvious - training score is much higher than cv score. But what is the reason for GradientBoost classifier?

zh_woo • Posted on Latest Version • 7 months ago • Options

2

Excellent idea, but i just got a score 0.7799, I do not know what happend....

Yassine Ghouz... • Posted on Latest Version • 7 months ago • Options

2

Hey!

You should get 0.79, since the kaggle kernel is limited to 20 min, i reduce the number of parameters in the grid search for the 5 algorithms for faster search. I obtained 0.81 with the optimal parameters. You should try with more parameters but it become slower (my prediction took me an 1h)

Yassine Ghouzam • Posted on Version 3 • 9 months ago • Options

4

Thanks, i think there is still a room for improvment without doing complicated things . For example , adding age categories ['child', 'teen', 'youngadult','medadult', 'old']. Since Age is difficult to impute, it maybe better to consider age categories, that will be much easier to predict with a multiclass classifier. Moreover, the observation of age distribution in survived and not survived subpop, clearly indicates categories of age with more or less chance to survived.

If you have any other ideas, your welcome :)

hptphuong • Posted on Latest Version • 8 months ago • Options

1

Could I ask a stupid question? Why you have to plot curve of feature important? I don't see you use any result from it. I really doesn't know why we need this step. Please help me explain.

Yassine Ghouz... • Posted on Latest Version • 8 months ago • Options

1

Feature importance histograms are indicative results, it was only for analysis purposes. It show that the prediction of the survival seems to be more associated with the Age, the Sex, the family size and the social standing of the passengers more than the location in the boat.

We could also use this to focus on the most relevant features for further analysis and modeling.

Rsanchez • Posted on Latest Version • 8 months ago • Options 1

Hello. Thanks for this nice kernel, it is a really good starting point for all people who are new in Kaggle. When looking for outliers you are just considering deviations in each variable individually. Have you ever tried more complicated approaches for outliers detection such as LOF? Thanks.

Yassine Ghouz... • Posted on Latest Version • 8 months ago • Options 0

Hey !

No i didnt , since we have only a few number of outliers, i wanted something easy to detect them . But thank you for the advice ill try .

Angel García Me... • Posted on Latest Version • 8 months ago • Options 1

Excellent. A very interesting analysis. Thanks for sharing.

phantom • Posted on Version 5 • 8 months ago • Options 1

I think I have found the very solution, Thank you!

Thiago Ribeiro • Posted on Version 5 • 8 months ago • Options 1

Excellent Kernel! Thanks for sharing!

Ivan Laskov • Posted on Latest Version • 8 months ago • Options 2

Good one.

NingLee • Posted on Version 4 • 9 months ago • Options 1

How much scores do you get after submiting? I run your kernel without modified and submit the result , but I only get scores 0.789... Am I doing something wrong? Plz help me, thx a lot

Yassine Ghouz... • Posted on Version 4 • 9 months ago • Options 2

Hey!

Yes its normal, since the kaggle kernel is limited to 20 min, i reduce the number of parameters in the grid search for the 5 algorithms for faster search. So the you probably miss the optimal parameters i used. You should try with more parameters but it become slower (my prediction took me an 1h)

Nithish Reddy • Posted on Version 4 • 9 months ago • Options 1

Excellent

Paul Larmuseau • Posted on Version 4 • 9 months ago • Options 1

One remark: if i join the train test and check then the outlier.... i don't find any...

Jinil C Sasidharan • Posted on Version 4 • 9 months ago • Options 1

This is really interesting.. I can learn a lot from this. Thank you

Sagarnil Das • Posted on Version 3 • 9 months ago • Options 1

Awesome man!Very helpful

Fukuball Lin • Posted on Version 3 • 9 months ago • Options 1

Impressive, I'll try your idea, thanks!

Abdulk • Posted on Version 3 • 9 months ago • Options 1

That's a very impressive kernel, a lot of interesting ideas, thank you Yassine.

IGNER • Posted on Version 3 • 9 months ago • Options 0

blabla

IGNER • Posted on Version 3 • 9 months ago • Options 0

..

Abdulk • Posted on Version 3 • 9 months ago • Options 0

blublu?

Sagarnil Das • Posted on Version 3 • 9 months ago • Options

0

Awesome man!Very helpful

Zsolt Kovacs • Posted on Version 4 • 9 months ago • Options

0

Interesting. Still reading through it, but one thing caught my attention: Why are you including all dummy variables? Shouldn't you use drop_first=True? (Ex in case of PClass, only have PClass2 and PClass3 included in the model) If not, why?

Yassine Ghouz... • Posted on Version 4 • 9 months ago • Options

6

Hey Zsolt !

Since perfect collinearity can be a problem, the suggested approach is to set drop_first to True to get n-1 columns. You should use drop_first = True when you have a regression model, perfect collinearities are very problematic leading to unreliable and unstable estimates of regression coefficients, more than for classification problems.

Multi-collinearity will not be a problem for certain classification models. Such as RF , DTC, gradient boosting. Different tree model will have different deal method, such as the random forest will keep them both (because they build the trees independently, and perform random selection of features for every trees). So the model will still work well.

In our case (for variables with 3 or more categories) multicollinearities can be safely ignored. Dont use categorical conversion for 2 categories variables (Ex: Sex).

Yassine Ghouzam

Zsolt Kovacs • Posted on Version 4 • 9 months ago • Options

0

Thanks! Appreciate the great explanation :)

Andrew Etchell • Posted on Latest Version • 3 months ago • Options

0

Thanks for posting the kernel! It was very helpful and really well explained. I'm totally new to machine learning and Kaggle etc, but initially had a bit of trouble reproducing the results. The code kept churning out an error until I added a line to the effect of dataset = pd.get_dummies(dataset, columns = ["Cabin"], prefix="Cab_"). From there, it seemed to work fine. Apologies if I've missed something very obvious, but I'm just wondering whether you intentionally didn't one hot encode cabin. Thanks again!

Leonardo Bartz • Posted on Version 4 • 9 months ago • Options

0

I'm using your analysis for studying, thanks!

Zsolt Kovacs • Posted on Version 4 • 9 months ago • Options 0

Hey, I'm back with some more questions:

1. As someone already pointed it out, if you check for outliers on the combined set, there aren't any. How come you decided to drop the outliers? (Yes, dropping them did cause my model's accuracy to go up with 0.02, but I'm wondering about your decision process)
2. Why did you decide to use those specific algorithms for the ensemble model? MLP and LogReg are also performing really well, even better than some of the ones you picked
3. Is there any value in dropping/grouping up the lowest performing features besides training speed? Ex: Would it actually improve the model if some ticket groups would be joined together?

An idea I thought of for improving the model would be to extract the surnames and look for spouses. Compare the probability of survival of couples vs parent+childer (Fsize=2) vs couple+other family (Fsize>2)

mgood2 • Posted on Version 4 • 9 months ago • Options 0

very helpful

RogerioFragoso • Posted on Version 4 • 9 months ago • Options 0

Very helpful kernel. Thank's a lot, Yassine Ghouzam.

Srinivasan Kanni... • Posted on Version 4 • 9 months ago • Options 0

This is useful for me!

Yohanes Gultom • Posted on Version 5 • 8 months ago • Options 0

This kernel helps me to understand how to analyze a dataset using Python. Thanks for sharing! I also have a question if you don't mind: how significant is the improvement of using VotingClassifier compared to the best individual Classifier (based on the learning rate, I think it's SVC)?

Yassine Ghouz... • Posted on Version 5 • 8 months ago • Options 1

Hey!

Ensemble methods (with a majority vote for example) perform generally better than individual classifiers, especially for classification problems more than for regression problems.

I get ~ 2%-3% improvements using the combinaison of 5 individuals classifiers compared to the RF classifier.

To get more familiar with ensemble learning and why it works better for classification problems, i invited you to read this excellent post : https://mlwave.com/kaggle-ensembling-guide/

Zhi Fei • Posted on Version 5 • 8 months ago • Options 0

Thank you

Yassine Ghouzam • Posted on Version 5 • 8 months ago • Options 0

Thank you for your comments ;)

Kevin1023 • Posted on Latest Version • 8 months ago • Options 0

Excellent !

AliceSmith • Posted on Latest Version • 8 months ago • Options 0

Great work!

ALittleMiss • Posted on Latest Version • 8 months ago • Options 0

Although It's a little difficult for me to read English Notebook.However,It's a really good analysis for me.I learn a lot .Thank you very much.

Boris Kalinin • Posted on Latest Version • 8 months ago • Options 0

Thank you very much for this kernel. But I didn't get, what 'To 4%' meant? Was it providing a top-4% score in Kaggle competitions?

Yassine Ghouz... • Posted on Latest Version • 8 months ago • Options 0

Hi Boris , Top 4% is the position in the leaderboard got with this code (with more hyperparameter tunning) in this compétition (for example you reach the 250th position over 7000 participants, you are in the top 4% of the leaderboard).

Artist • Posted on Latest Version • 8 months ago • Options 0

great!

Tanmoy • Posted on Latest Version • 8 months ago • Options 0

/opt/conda/lib/python3.6/site-packages/numpy/lib/function_base.py:4269: RuntimeWarning: Invalid value encountered in percentile interpolation=interpolation)

Percentile calculation in step 3 throws RunTimeWarning

Rafal Plis • Posted on Latest Version • 8 months ago • Options 0

~~Excellent explanation and pretty graphs. Thanks. Very useful for me.~~

saddam salah • Posted on Latest Version • 7 months ago • Options 0

Excellent

Nimrod Daniel • Posted on Latest Version • 7 months ago • Options 0

Great job! :)

Kristofer Rolf Söd... • Posted on Latest Version • 7 months ago • Options 0

Thank you for sharing!

LilyChen • Posted on Latest Version • 6 months ago • Options 0

Thanks for sharing!

sion. • Posted on Latest Version • 6 months ago • Options 0

Thanks for great study!!

Marsh • Posted on Latest Version • 6 months ago • Options 0

Thank you for sharing this Yassine. I've learned a ton by studying this excellent kernel.

Ramona Fli • Posted on Latest Version • 6 months ago • Options 0

Thanks for your kernel! very helpful! any idea how much the classifiers could be correlated in order to have a meaningful ensemble?

y.n.v.vikas • Posted on Latest Version • 6 months ago • Options 0

Explanation is excellent :)

kunimune • Posted on Latest Version • 6 months ago • Options 0

Thank you for your sharing.

Orix Au Yeung • Posted on Latest Version • 6 months ago • Options 0

Thank you for you great work! I'm a newcomer in ML and this notebook has provided me many invaluable practical insights.

S Francis • Posted on Latest Version • 5 months ago • Options 0

Thanks for taking the time to write this up - really helpful for a newcomer.

Dhealthy • Posted on Latest Version • 5 months ago • Options 0

Dhealthy • Posted on Latest Version • 5 months ago • Options 0

Shadab • Posted on Latest Version • 5 months ago • Options 0

1 thing that is bugging me is that i followed every step but made some minor changes in filling missinge values of Age and categorizing it. Used RandomForestClassifier with GridSearch and got best accuracy of 1.0. Howeven when submitted my answer, i scored 49%. Is it even possible or is it overfitting/undefitting?

AbhishekMam... • Posted on Latest Version • 5 months ago • Options 0

This is because of over fitting. The accuracy score you are getting is based on your training data. However, the score you get after submitting is scored on a new test data. So since your model works good on only your train data means that the model is biased and has been over fitted. you can try using k-fold training method, it will reduce the bias.

Shadab • Posted on Latest Version • 5 months ago • Options 0

thanks, that helped

Priyansh Agarwal • Posted on Latest Version • 5 months ago • Options 0

Great Work...

Pratik Singh • Posted on Latest Version • 5 months ago • Options 0

```
index_NaN_age = list(train_df["Age"][train_df["Age"].isnull()].index)

for i in index_NaN_age:
age_med = train_df["Age"].median()
    age_pred = train_df["Age"][((train_df['SibSp'] == train_df.iloc[i]["SibSp"]) &
                                (train_df['Parch'] == train_df.iloc[i]["Parch"]) &
```

```
                          (train_df['Pclass'] == train_df.iloc[i]["Pclass"]))].median()
    print(i)
    print(age_pred)
```

Hi , In this above setting , I didn't get the logic behind the if else clause.Moreover the complete list of age_pred gives

158 26.0 159 nan 166 37.5 168

showing index 159 to satisfy the else clause.but the complete dataset shows a different result.

```
i=159
train_df['Age'][((train_df['SibSp'] == train_df.iloc[i]["SibSp"]) &amp;
                 (train_df['Parch'] == train_df.iloc[i]["Parch"]) &amp;
                 (train_df['Pclass'] == train_df.iloc[i]["Pclass"]))]
```

159 28.0 180 28.0 201 28.0 324 28.0 792 28.0 846 28.0 863 28.0

fuqiuai • Posted on Latest Version • 4 months ago • Options 0

Thank you, it's really very helpful!

emmanuelmance... • Posted on Latest Version • 4 months ago • Options 0

Great job ! Very informative and instructive.

Alex Blaer • Posted on Latest Version • 4 months ago • Options 0

Really helpful. Thanks a lot.

Robert Lowry • Posted on Latest Version • 4 months ago • Options 0

Thank you for a wonderful kernel! I learned a lot!

llemonm • Posted on Latest Version • 4 months ago • Options 0

Hello, Can I ask you a question? After I tuned the parameters for the five classifiers, their cv score is 0.83 (without tuning the parameters is 0.81). But I didn't get improvement in the final LB score, it remains the same(0.80). Why this happens? It really confuse me, I really hope your answer.

Leonardo Ferre... • Posted on Latest Version • 3 months ago • Options 0

I have the same question! I am stuck on .7990 and my cv = Kfold(10) tells me accuracy of .81 ~.83 but when I submit to Kaggle this show me .77 ~ .79

Richard Ackon • Posted on Latest Version • 3 months ago • Options 0

Excellent work!

Richard Ackon • Posted on Latest Version • 3 months ago • Options

0

Andre Koeppl • Posted on Latest Version • 3 months ago • Options

0

Great Kernel, thanks!

kailangdebo • Posted on Latest Version • 3 months ago • Options

0

it'great.thankyou yassine

Aleks Wittkamp • Posted on Latest Version • 2 months ago • Options

0

Great kernel. I learned a lot from the way you explored the data. Thanks.

Abhishek Singh • Posted on Latest Version • 2 months ago • Options

0

Lovely..Great Work..Do you have any kernel for anomaly detection and text mining.. Please suggest

Thawatchai Rang... • Posted on Latest Version • 2 months ago • Options

0

I learned a lot in this kernel. Very organized, thank you

Carey B • Posted on Latest Version • 2 months ago • Options

0

Good Kernel. Clear and informative. It has been very enlightening seeing all the different styles of Kernels for just the Titanic data set.

Stephen PU • Posted on Latest Version • 2 months ago • Options

0

Thanks, Excellent!!

Lana Samoilova • Posted on Latest Version • 2 months ago • Options

0

It is an excellent explanation. Thank you!

Lukasz Zawieska • Posted on Latest Version • 2 months ago • Options

0

Amazing job ! Thanks for sharing ! :)

Prabhjot Kaur • Posted on Latest Version • a month ago • Options 0

Great kernel, Thanks.

Esteban Cortero • Posted on Latest Version • a month ago • Options 0

Excellent notebook! So far my favorite Titanic kernel! Any reason why you didn't use sklearn's built-in outlier detection instead of implementing your own?

Graziano Ucci • Posted on Latest Version • a month ago • Options 0

Great job, thanks!

FlyFish2018 • Posted on Latest Version • 22 days ago • Options 0

very good.Thank you for your work.

panpolikarp • Posted on Latest Version • 20 days ago • Options 0

Very helpfull stuff! :)

xuyiren • Posted on Latest Version • 18 days ago • Options 0

Good Job! Thank you!

Nick Braunagel • Posted on Latest Version • 16 days ago • Options 0

For those who go through this very good notebook (thank you, Yassine Ghouzam) and find their CV-tuned model scores in the low 80%'s but later (post-submission to Kaggle) receives a lower score (high 70%'s), be sure to investigate over-fitting.

The model you built is likely over-fitting the training data at the expense of being less accurate when predicting on the test data (i.e. not well *generalized*). Here is a short Q&A about this issue and approaches to deal with it. This Q&A even addresses Kaggle competitions:

> Another example where [over-fitting] does occur is in machine learning competitions with a leader-board based on a validation set. Inevitably some competitors keep tinkering with their model to get further up the leader board, but then end up towards the bottom of the final rankings. The reason for this is that their [models] have over-fitted the validation set (effectively learning the random variations in the small validation set).

There are also lots of good discussions on Kaggle about dealing with over-fitting due to model CV-tuning.

freepy • Posted on Latest Version • 13 days ago • Options 0

An amazing kernel that shows a well thought-out workflow, as well as results! Thank you so much for sharing!

平 赵 • Posted on Latest Version • 13 days ago • Options

0

Thank you very much

JubaerHossain • Posted on Latest Version • 10 days ago • Options

0

Great!!!

YiLuo • Posted on Latest Version • 8 days ago • Options

0

Thanks for this great Kernel. Really learned a lot from it. Just one question, I used similar feature engineering and `RandomForestClassifier()` , in the 5-fold cv, it achieves 0.8316498316498316 accuracy. However, when submit the final prediction, it only achieves 78.3%. Any idea what causes it?

Georgios Sara... • Posted on Latest Version • 8 days ago • Options

0

It probably means that your current model is overfitted to the 5 data folds of the train set that you have used for training your model so far. It lacks the ability to generalize to new data, which Kaggle reserves for testing your model. The best thing to do would be to leave a validation set outside the model (e.g. 20% of all the data), run the k-fold cross-validation to build your model and then check its performance on the validation/hold-out set. This number should be close to what you get when you upload your results on Kaggle's test set.

Akinwande Ko... • Posted on Latest Version • 3 days ago • Options

0

The best thing to do would be to leave a validation set outside the model (e.g. 20% of all the data), run the k-fold cross-validation to build your model. Please explain better

Georgios Sara... • Posted on Latest Version • 2 days ago • Options

0

Lets say your data set has 100 rows. You should not use all the data to train a model from the beginning. You start by taking a random sample of these (80% for example) and leaving the rest outside of the model (this 20% shall be called development set). You train a model (whichever model, it can be Boosted Trees, Neural Nets, etc) using k-fold cross validation to obtain the best hyper-parameters for the model. When you are happy with the initial model you predict on the development set. You check the selected metrics on this prediction (since you know its true value). If you are happy again you proceed to predict on the Public leaderboard test set. What you must watch now is to build a model that can generalize both in your development data set and on the public leaderboard data set. If it performs well only on one of these sets, it means that your model is overfitted to this set and it will not be able to perform well on Private Leaderboard. Keep this in mind: your goal is to build a model with the least variation in the selected loss function between the development set and public leaderboard set.

Anto Ponselvan • Posted on Latest Version • 5 days ago • Options

0

Thank you very much. Was very useful for a beginner like me. :)

Murat Can ALPAY • Posted on Latest Version • 6 hours ago • Options

0

Thanks, I now have better understanding of how to use voting