Manav Sehgal

Titanic Data Science Solutions

last run 3 months ago · Python notebook · 41075 views
using data from Titanic: Machine Learning from Disaster · Public

Titanic Data Science Solutions

This notebook is companion to the book Data Science Solutions (https://startupsci.com). The notebook walks us through a typical workflow for solving data science competitions at sites like Kaggle.

There are several excellent notebooks to study data science competition entries. However many will skip some of the explanation on how the solution is developed as these notebooks are developed by experts for experts. The objective of this notebook is to follow a step-by-step workflow, explaining each step and rationale for every decision we take during solution development.

Workflow stages

The competition solution workflow goes through seven stages described in the Data Science Solutions book.

1. Question or problem definition.
2. Acquire training and testing data.
3. Wrangle, prepare, cleanse the data.
4. Analyze, identify patterns, and explore the data.
5. Model, predict and solve the problem.
6. Visualize, report, and present the problem solving steps and final solution.
7. Supply or submit the results.

The workflow indicates general sequence of how each stage may follow the other. However there are use cases with exceptions.

- We may combine mulitple workflow stages. We may analyze by visualizing data.
- Perform a stage earlier than indicated. We may analyze data before and after wrangling.
- Perform a stage multiple times in our workflow. Visualize stage may be used multiple times.
- Drop a stage altogether. We may not need supply stage to productize or service enable our dataset for a competition.

Question and problem definition

Competition sites like Kaggle define the problem to solve or questions to ask while providing the datasets for training your data science model and testing the model results against a test dataset. The question or problem definition for Titanic Survival competition is described here at Kaggle (https://www.kaggle.com/c/titanic).

Survival competition is described here at Kaggle (https://www.kaggle.com/c/titanic).

> Knowing from a training set of samples listing passengers who survived or did not survive the Titanic disaster, can our model determine based on a given test dataset not containing the survival information, if these passengers in the test dataset survived or not.

We may also want to develop some early understanding about the domain of our problem. This is described on the Kaggle competition description page here (https://www.kaggle.com/c/titanic). Here are the highlights to note.

- On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. Translated 32% survival rate.
- One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew.
- Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

Workflow goals

The data science solutions workflow solves for seven major goals.

Classifying. We may want to classify or categorize our samples. We may also want to understand the implications or correlation of different classes with our solution goal.

Correlating. One can approach the problem based on available features within the training dataset. Which features within the dataset contribute significantly to our solution goal? Statistically speaking is there a correlation (https://en.wikiversity.org/wiki/Correlation) among a feature and solution goal? As the feature values change does the solution state change as well, and visa-versa? This can be tested both for numerical and categorical features in the given dataset. We may also want to determine correlation among features other than survival for subsequent goals and workflow stages. Correlating certain features may help in creating, completing, or correcting features.

Converting. For modeling stage, one needs to prepare the data. Depending on the choice of model algorithm one may require all features to be converted to numerical equivalent values. So for instance converting text categorical values to numeric values.

Completing. Data preparation may also require us to estimate any missing values within a feature. Model algorithms may work best when there are no missing values.

Correcting. We may also analyze the given training dataset for errors or possibly innacurate values within features and try to corrent these values or exclude the samples containing the errors. One way to do this is to detect any outliers among our samples or features. We may also completely discard a feature if it is not contribting to the analysis or may significantly skew the results.

Creating. Can we create new features based on an existing feature or a set of features, such that the new feature follows the correlation, conversion, completeness goals.

Charting. How to select the right visualization plots and charts depending on nature of the data and the solution goals. A good start is to read the Tableau paper on Which chart or graph is right for you? (http://www.tableau.com/learn/whitepapers/which-chart-or-graph-is-right-for-you#ERAcoH5sEG5CFlek.99).

Refactor Release 2017-Jan-29

We are significantly refactoring the notebook based on (a) comments received by readers, (b) issues in porting notebook from Jupyter kernel (2.7) to Kaggle kernel (3.5), and (c) review of few more best practice kernels.

User comments

- Combine training and test data for certain operations like converting titles across dataset to numerical values. (thanks @Sharan Naribole)
- Correct observation - nearly 30% of the passengers had siblings and/or spouses aboard. (thanks @Reinhard)

- Correctly interpreting logistic regresssion coefficients. (thanks @Reinhard)

Porting issues

- Specify plot dimensions, bring legend into plot.

Best practices

- Performing feature correlation analysis early in the project.
- Using multiple plots instead of overlays for readability.

In [1]:
```
# data analysis and wrangling
import pandas as pd
import numpy as np
import random as rnd

# visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# machine learning
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.tree import DecisionTreeClassifier
```

Acquire data

The Python Pandas packages helps us work with our datasets. We start by acquiring the training and testing datasets into Pandas DataFrames. We also combine these datasets to run certain operations on both datasets together.

In [2]:
```
train_df = pd.read_csv('../input/train.csv')
test_df = pd.read_csv('../input/test.csv')
combine = [train_df, test_df]
```

Analyze by describing data

Pandas also helps describe the datasets answering following questions early in our project.

Which features are available in the dataset?

Noting the feature names for directly manipulating or analyzing these. These feature names are described on the Kaggle data page here (https://www.kaggle.com/c/titanic/data).

In [3]:
```
print(train_df.columns.values)
```

```
['PassengerId' 'Survived' 'Pclass' 'Name' 'Sex' 'Age' 'SibSp' 'Parch'
 'Ticket' 'Fare' 'Cabin' 'Embarked']
```

Which features are categorical?

These values classify the samples into sets of similar samples. Within categorical features are the values nominal, ordinal, ratio, or interval based? Among other things this helps us select the appropriate plots for visualization.

- Categorical: Survived, Sex, and Embarked. Ordinal: Pclass.

Which features are numerical?

Which features are numerical? These values change from sample to sample. Within numerical features are the values discrete, continuous, or timeseries based? Among other things this helps us select the appropriate plots for visualization.

- Continous: Age, Fare. Discrete: SibSp, Parch.

In [4]:
```
# preview the data
train_df.head()
```

Out[4]:

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

Which features are mixed data types?

Numerical, alphanumeric data within same feature. These are candidates for correcting goal.

- Ticket is a mix of numeric and alphanumeric data types. Cabin is alphanumeric.

Which features may contain errors or typos?

This is harder to review for a large dataset, however reviewing a few samples from a smaller dataset may just tell us outright, which features may require correcting.

- Name feature may contain errors or typos as there are several ways used to describe a name including titles, round brackets, and quotes used for alternative or short names.

In [5]:
```
train_df.tail()
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emb
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
				Graham								

887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

Which features contain blank, null or empty values?

These will require correcting.

- Cabin > Age > Embarked features contain a number of null values in that order for the training dataset.
- Cabin > Age are incomplete in case of test dataset.

What are the data types for various features?

Helping us during converting goal.

- Seven features are integer or floats. Six in case of test dataset.
- Five features are strings (object).

In [6]:
```
train_df.info()
print('_'*40)
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
________________________________________
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Name           418 non-null object
Sex            418 non-null object
Age            332 non-null float64
SibSp          418 non-null int64
```

```
Parch          418 non-null int64
Ticket         418 non-null object
Fare           417 non-null float64
Cabin          91 non-null object
Embarked       418 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

What is the distribution of numerical feature values across the samples?

This helps us determine, among other early insights, how representative is the training dataset of the actual problem domain.

- Total samples are 891 or 40% of the actual number of passengers on board the Titanic (2,224).
- Survived is a categorical feature with 0 or 1 values.
- Around 38% samples survived representative of the actual survival rate at 32%.
- Most passengers (> 75%) did not travel with parents or children.
- Nearly 30% of the passengers had siblings and/or spouse aboard.
- Fares varied significantly with few passengers (<1%) paying as high as $512.
- Few elderly passengers (<1%) within age range 65-80.

In [7]:
```
train_df.describe()
# Review survived rate using `percentiles=[.61, .62]` knowing our problem descript
ion mentions 38% survival rate.
# Review Parch distribution using `percentiles=[.75, .8]`
# SibSp distribution `[.68, .69]`
# Age and Fare `[.1, .2, .3, .4, .5, .6, .7, .8, .9, .99]`
```

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

What is the distribution of categorical features?

- Names are unique across the dataset (count=unique=891)
- Sex variable as two possible values with 65% male (top=male, freq=577/count=891).
- Cabin values have several dupicates across samples. Alternatively several passengers shared a cabin.
- Embarked takes three possible values. S port used by most passengers (top=S)
- Ticket feature has high ratio (22%) of duplicate values (unique=681).

In [8]:
```
train_df.describe(include=['O'])
```

Out[8]:

	Name	Sex	Ticket	Cabin	Embarked
count	891	891	891	204	889
unique	891	2	681	147	3
top	Chronopoulos, Mr. Apostolos	male	CA. 2343	G6	S
freq	1	577	7	4	644

Assumtions based on data analysis

We arrive at following assumptions based on data analysis done so far. We may validate these assumptions further before taking appropriate actions.

Correlating.

We want to know how well does each feature correlate with Survival. We want to do this early in our project and match these quick correlations with modelled correlations later in the project.

Completing.

1. We may want to complete Age feature as it is definitely correlated to survival.
2. We may want to complete the Embarked feature as it may also correlate with survival or another important feature.

Correcting.

1. Ticket feature may be dropped from our analysis as it contains high ratio of duplicates (22%) and there may not be a correlation between Ticket and survival.
2. Cabin feature may be dropped as it is highly incomplete or contains many null values both in training and test dataset.
3. PassengerId may be dropped from training dataset as it does not contribute to survival.
4. Name feature is relatively non-standard, may not contribute directly to survival, so maybe dropped.

Creating.

1. We may want to create a new feature called Family based on Parch and SibSp to get total count of family members on board.
2. We may want to engineer the Name feature to extract Title as a new feature.
3. We may want to create new feature for Age bands. This turns a continous numerical feature into an ordinal categorical feature.
4. We may also want to create a Fare range feature if it helps our analysis.

Classifying.

We may also add to our assumptions based on the problem description noted earlier.

1. Women (Sex=female) were more likely to have survived.
2. Children (Age<?) were more likely to have survived.
3. The upper-class passengers (Pclass=1) were more likely to have survived.

Analyze by pivoting features

To confirm some of our observations and assumptions, we can quickly analyze our feature correlations by pivoting features against each other. We can only do so at this stage for features which do not have any empty values. It also makes sense doing so only for features which are categorical (Sex), ordinal (Pclass) or discrete (SibSp, Parch) type.

- **Pclass** We observe significant correlation (>0.5) among Pclass=1 and Survived (classifying #3). We decide to include this feature in our model.
- **Sex** We confirm the observation during problem definition that Sex=female had very high survival rate at 74% (classifying #1).
- **SibSp and Parch** These features have zero correlation for certain values. It may be best to derive a feature or a set of features from these individual features (creating #1).

In [9]:
```
train_df[['Pclass', 'Survived']].groupby(['Pclass'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

Out[9]:

	Pclass	Survived
0	1	0.629630

0	1	0.629630
1	2	0.472826
2	3	0.242363

In [10]:
```
train_df[["Sex", "Survived"]].groupby(['Sex'],
as_index=False).mean().sort_values(by='Survived', ascending=False)
```

Out[10]:

	Sex	Survived
0	female	0.742038
1	male	0.188908

In [11]:
```
train_df[["SibSp", "Survived"]].groupby(['SibSp'], as_index=False).mean().sort_val
ues(by='Survived', ascending=False)
```

Out[11]:

	SibSp	Survived
1	1	0.535885
2	2	0.464286
0	0	0.345395
3	3	0.250000
4	4	0.166667
5	5	0.000000
6	8	0.000000

In [12]:
```
train_df[["Parch", "Survived"]].groupby(['Parch'], as_index=False).mean().sort_val
ues(by='Survived', ascending=False)
```

Out[12]:

	Parch	Survived
3	3	0.600000
1	1	0.550847
2	2	0.500000
0	0	0.343658
5	5	0.200000
4	4	0.000000
6	6	0.000000

Analyze by visualizing data

Now we can continue confirming some of our assumptions using visualizations for analyzing the data.

Correlating numerical features

Let us start by understanding correlations between numerical features and our solution goal (Survived).

A histogram chart is useful for analyzing continous numerical variables like Age where banding or ranges will help identify useful patterns. The histogram can indicate distribution of samples using automatically defined bins or equally ranged bands. This helps us answer questions relating to specific bands (Did infants have better survival rate?)

Note that x-axis in historgram visualizations represents the count of samples or passengers.

Observations.

- Infants (Age <=4) had high survival rate.
- Oldest passengers (Age = 80) survived.
- Large number of 15-25 year olds did not survive.
- Most passengers are in 15-35 age range.

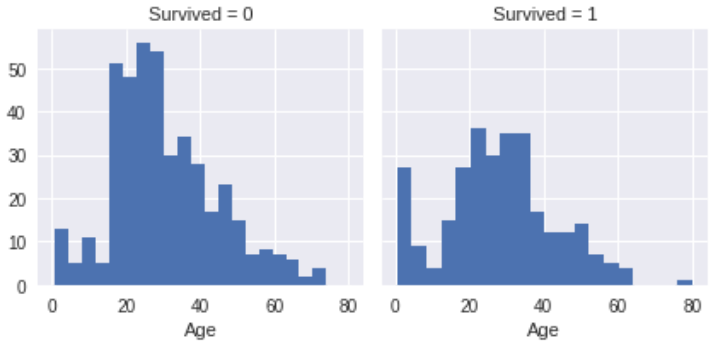
Decisions.

This simple analysis confirms our assumptions as decisions for subsequent workflow stages.

- We should consider Age (our assumption classifying #2) in our model training.
- Complete the Age feature for null values (completing #1).
- We should band age groups (creating #3).

In [13]:
```
g = sns.FacetGrid(train_df, col='Survived')
g.map(plt.hist, 'Age', bins=20)
```

Out[13]: <seaborn.axisgrid.FacetGrid at 0x7ff74ffd3f98>



Correlating numerical and ordinal features

We can combine multiple features for identifying correlations using a single plot. This can be done with numerical and categorical features which have numeric values.

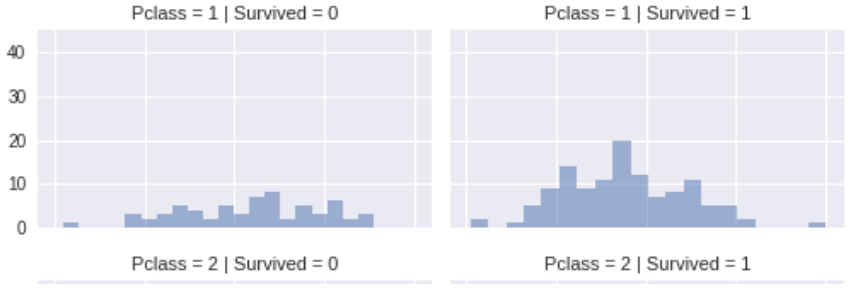
Observations.

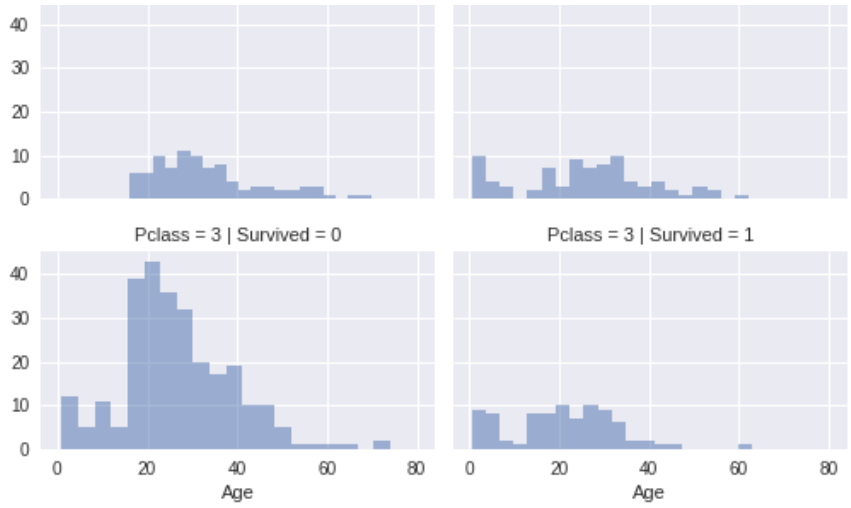
- Pclass=3 had most passengers, however most did not survive. Confirms our classifying assumption #2.
- Infant passengers in Pclass=2 and Pclass=3 mostly survived. Further qualifies our classifying assumption #2.
- Most passengers in Pclass=1 survived. Confirms our classifying assumption #3.
- Pclass varies in terms of Age distribution of passengers.

Decisions.

- Consider Pclass for model training.

In [14]:
```
# grid = sns.FacetGrid(train_df, col='Pclass', hue='Survived')
grid = sns.FacetGrid(train_df, col='Survived', row='Pclass', size=2.2, aspect=1.6)
grid.map(plt.hist, 'Age', alpha=.5, bins=20)
grid.add_legend();
```





Correlating categorical features

Now we can correlate categorical features with our solution goal.

Observations.

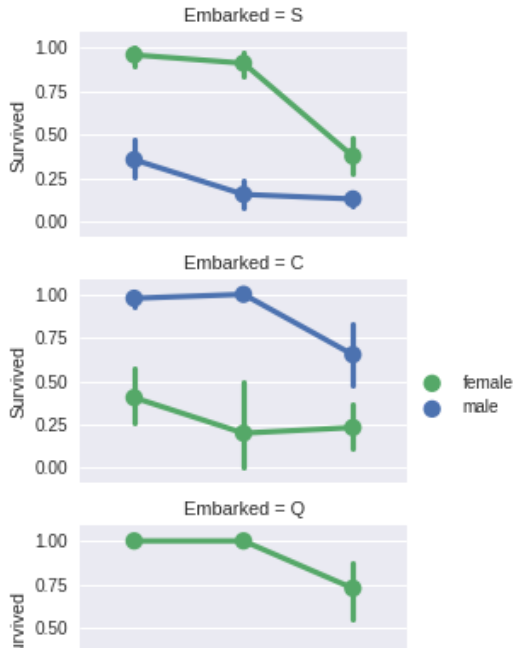
- Female passengers had much better survival rate than males. Confirms classifying (#1).
- Exception in Embarked=C where males had higher survival rate. This could be a correlation between Pclass and Embarked and in turn Pclass and Survived, not necessarily direct correlation between Embarked and Survived.
- Males had better survival rate in Pclass=3 when compared with Pclass=2 for C and Q ports. Completing (#2).
- Ports of embarkation have varying survival rates for Pclass=3 and among male passengers. Correlating (#1).

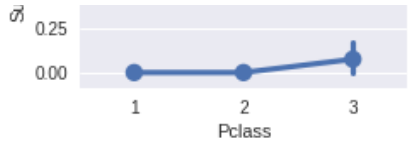
Decisions.

- Add Sex feature to model training.
- Complete and add Embarked feature to model training.

In [15]:
```
# grid = sns.FacetGrid(train_df, col='Embarked')
grid = sns.FacetGrid(train_df, row='Embarked', size=2.2, aspect=1.6)
grid.map(sns.pointplot, 'Pclass', 'Survived', 'Sex', palette='deep')
grid.add_legend()
```

Out[15]: <seaborn.axisgrid.FacetGrid at 0x7ff74c535ba8>





Correlating categorical and numerical features

We may also want to correlate categorical features (with non-numeric values) and numeric features. We can consider correlating Embarked (Categorical non-numeric), Sex (Categorical non-numeric), Fare (Numeric continuous), with Survived (Categorical numeric).

Observations.

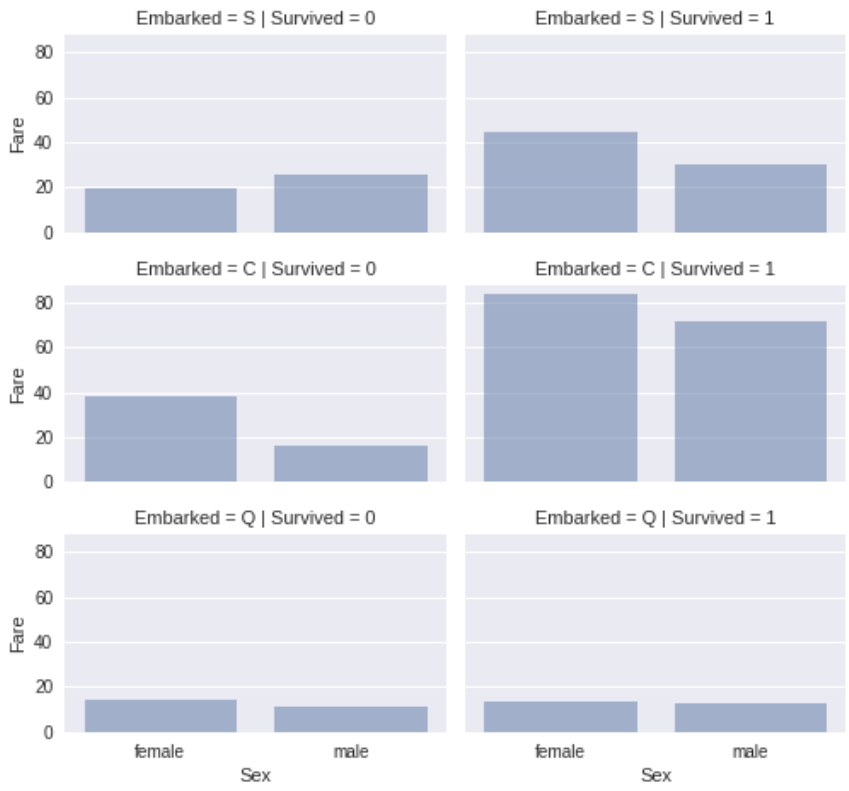
- Higher fare paying passengers had better survival. Confirms our assumption for creating (#4) fare ranges.
- Port of embarkation correlates with survival rates. Confirms correlating (#1) and completing (#2).

Decisions.

- Consider banding Fare feature.

In [16]:
```
# grid = sns.FacetGrid(train_df, col='Embarked', hue='Survived', palette={0: 'k',
 1: 'w'})
grid = sns.FacetGrid(train_df, row='Embarked', col='Survived', size=2.2, aspect=1.
6)
grid.map(sns.barplot, 'Sex', 'Fare', alpha=.5, ci=None)
grid.add_legend()
```

Out[16]: <seaborn.axisgrid.FacetGrid at 0x7ff74c3df780>



Wrangle data

We have collected several assumptions and decisions regarding our datasets and solution requirements. So far we did not have to change a single feature or value to arrive at these. Let us now execute our decisions and assumptions for correcting,

creating, and completing goals.

Correcting by dropping features

This is a good starting goal to execute. By dropping features we are dealing with fewer data points. Speeds up our notebook and eases the analysis.

Based on our assumptions and decisions we want to drop the Cabin (correcting #2) and Ticket (correcting #1) features.

Note that where applicable we perform operations on both training and testing datasets together to stay consistent.

In [17]:
```
print("Before", train_df.shape, test_df.shape, combine[0].shape, combine[1].shape)

train_df = train_df.drop(['Ticket', 'Cabin'], axis=1)
test_df = test_df.drop(['Ticket', 'Cabin'], axis=1)
combine = [train_df, test_df]

"After", train_df.shape, test_df.shape, combine[0].shape, combine[1].shape
```

```
Before (891, 12) (418, 11) (891, 12) (418, 11)
```

Out[17]:
```
('After', (891, 10), (418, 9), (891, 10), (418, 9))
```

Creating new feature extracting from existing

We want to analyze if Name feature can be engineered to extract titles and test correlation between titles and survival, before dropping Name and PassengerId features.

In the following code we extract Title feature using regular expressions. The RegEx pattern `(\w+\.)` matches the first word which ends with a dot character within Name feature. The `expand=False` flag returns a DataFrame.

Observations.

When we plot Title, Age, and Survived, we note the following observations.

- Most titles band Age groups accurately. For example: Master title has Age mean of 5 years.
- Survival among Title Age bands varies slightly.
- Certain titles mostly survived (Mme, Lady, Sir) or did not (Don, Rev, Jonkheer).

Decision.

- We decide to retain the new Title feature for model training.

In [18]:
```
for dataset in combine:
    dataset['Title'] = dataset.Name.str.extract(' ([A-Za-z]+)\.', expand=False)

pd.crosstab(train_df['Title'], train_df['Sex'])
```

Out[18]:

Sex	female	male
Title		
Capt	0	1
Col	0	2
Countess	1	0
Don	0	1
Dr	1	6
Jonkheer	0	1
Lady	1	0
Major	0	2

Major	0	2
Master	0	40
Miss	182	0
Mlle	2	0
Mme	1	0
Mr	0	517
Mrs	125	0
Ms	1	0
Rev	0	6
Sir	0	1

We can replace many titles with a more common name or classify them as `Rare`.

In [19]:
```
for dataset in combine:
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Capt',
'Col',\
        'Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')

    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')

train_df[['Title', 'Survived']].groupby(['Title'], as_index=False).mean()
```

Out[19]:

	Title	Survived
0	Master	0.575000
1	Miss	0.702703
2	Mr	0.156673
3	Mrs	0.793651
4	Rare	0.347826

We can convert the categorical titles to ordinal.

In [20]:
```
title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}
for dataset in combine:
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0)

train_df.head()
```

Out[20]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	7.2500	S	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	71.2833	C	3
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	7.9250	S	2

3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	53.1000	S	3
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	8.0500	S	1

Now we can safely drop the Name feature from training and testing datasets. We also do not need the PassengerId feature in the training dataset.

In [21]:
```
train_df = train_df.drop(['Name', 'PassengerId'], axis=1)
test_df = test_df.drop(['Name'], axis=1)
combine = [train_df, test_df]
train_df.shape, test_df.shape
```

Out[21]: ((891, 9), (418, 9))

Converting a categorical feature

Now we can convert features which contain strings to numerical values. This is required by most model algorithms. Doing so will also help us in achieving the feature completing goal.

Let us start by converting Sex feature to a new feature called Gender where female=1 and male=0.

In [22]:
```
for dataset in combine:
    dataset['Sex'] = dataset['Sex'].map( {'female': 1, 'male': 0} ).astype(int)

train_df.head()
```

Out[22]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	0	3	0	22.0	1	0	7.2500	S	1
1	1	1	1	38.0	1	0	71.2833	C	3
2	1	3	1	26.0	0	0	7.9250	S	2
3	1	1	1	35.0	1	0	53.1000	S	3
4	0	3	0	35.0	0	0	8.0500	S	1

Completing a numerical continuous feature

Now we should start estimating and completing features with missing or null values. We will first do this for the Age feature.

We can consider three methods to complete a numerical continuous feature.

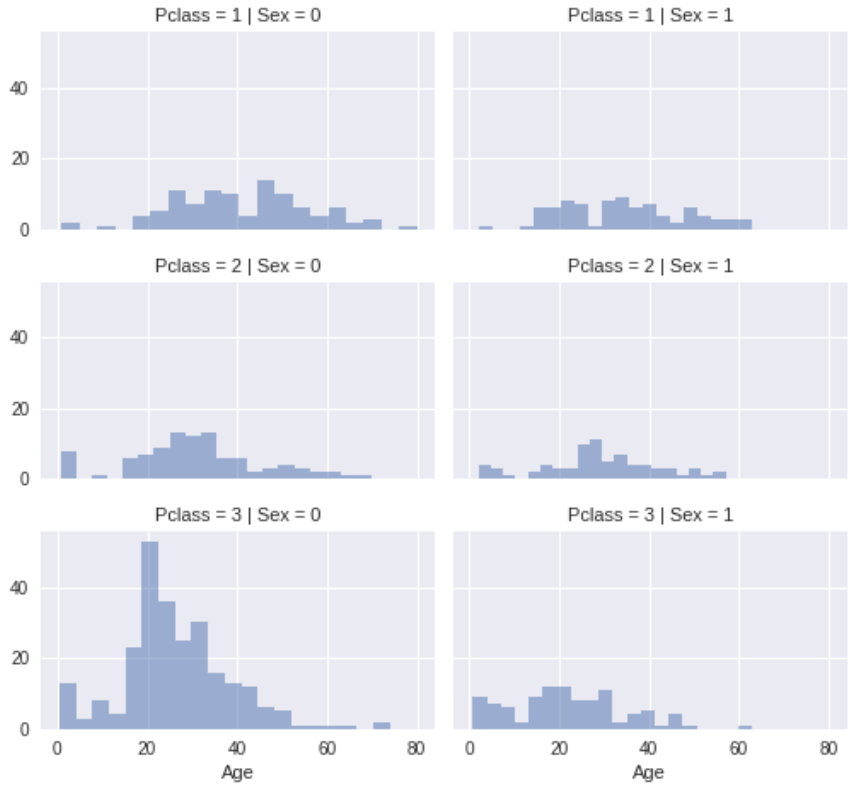
1. A simple way is to generate random numbers between mean and standard deviation (https://en.wikipedia.org/wiki/Standard_deviation).
2. More accurate way of guessing missing values is to use other correlated features. In our case we note correlation among Age, Gender, and Pclass. Guess Age values using median (https://en.wikipedia.org/wiki/Median) values for Age across sets of Pclass and Gender feature combinations. So, median Age for Pclass=1 and Gender=0, Pclass=1 and Gender=1, and so on...
3. Combine methods 1 and 2. So instead of guessing age values based on median, use random numbers between mean and standard deviation, based on sets of Pclass and Gender combinations.

Method 1 and 3 will introduce random noise into our models. The results from multiple executions might vary. We will prefer

method 2.

In [23]:
```
# grid = sns.FacetGrid(train_df, col='Pclass', hue='Gender')
grid = sns.FacetGrid(train_df, row='Pclass', col='Sex', size=2.2, aspect=1.6)
grid.map(plt.hist, 'Age', alpha=.5, bins=20)
grid.add_legend()
```

Out[23]: <seaborn.axisgrid.FacetGrid at 0x7ff74bfede80>



Let us start by preparing an empty array to contain guessed Age values based on Pclass x Gender combinations.

In [24]:
```
guess_ages = np.zeros((2,3))
guess_ages
```

Out[24]:
```
array([[ 0.,  0.,  0.],
       [ 0.,  0.,  0.]])
```

Now we iterate over Sex (0 or 1) and Pclass (1, 2, 3) to calculate guessed values of Age for the six combinations.

In [25]:
```
for dataset in combine:
    for i in range(0, 2):
        for j in range(0, 3):
            guess_df = dataset[(dataset['Sex'] == i) & \
                               (dataset['Pclass'] == j+1)]['Age'].dropna()

            # age_mean = guess_df.mean()
            # age_std = guess_df.std()
            # age_guess = rnd.uniform(age_mean - age_std, age_mean + age_std)

            age_guess = guess_df.median()

            # Convert random age float to nearest .5 age
            guess_ages[i,j] = int( age_guess/0.5 + 0.5 ) * 0.5

    for i in range(0, 2):
        for j in range(0, 3):
```

```
            dataset.loc[ (dataset.Age.isnull()) & (dataset.Sex == i) & (dataset.Pc
lass == j+1),\
                    'Age'] = guess_ages[i,j]

    dataset['Age'] = dataset['Age'].astype(int)

train_df.head()
```

Out[25]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	0	3	0	22	1	0	7.2500	S	1
1	1	1	1	38	1	0	71.2833	C	3
2	1	3	1	26	0	0	7.9250	S	2
3	1	1	1	35	1	0	53.1000	S	3
4	0	3	0	35	0	0	8.0500	S	1

Let us create Age bands and determine correlations with Survived.

In [26]:
```
train_df['AgeBand'] = pd.cut(train_df['Age'], 5)
train_df[['AgeBand', 'Survived']].groupby(['AgeBand'], as_index=False).mean().sort
_values(by='AgeBand', ascending=True)
```

Out[26]:

	AgeBand	Survived
0	(-0.08, 16]	0.550000
1	(16, 32]	0.337374
2	(32, 48]	0.412037
3	(48, 64]	0.434783
4	(64, 80]	0.090909

Let us replace Age with ordinals based on these bands.

In [27]:
```
for dataset in combine:
    dataset.loc[ dataset['Age'] <= 16, 'Age'] = 0
    dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 32), 'Age'] = 1
    dataset.loc[(dataset['Age'] > 32) & (dataset['Age'] <= 48), 'Age'] = 2
    dataset.loc[(dataset['Age'] > 48) & (dataset['Age'] <= 64), 'Age'] = 3
    dataset.loc[ dataset['Age'] > 64, 'Age']
train_df.head()
```

Out[27]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title	AgeBand
0	0	3	0	1	1	0	7.2500	S	1	(16, 32]
1	1	1	1	2	1	0	71.2833	C	3	(32, 48]
2	1	3	1	1	0	0	7.9250	S	2	(16, 32]
3	1	1	1	2	1	0	53.1000	S	3	(32, 48]
4	0	3	0	2	0	0	8.0500	S	1	(32, 48]

We can not remove the AgeBand feature.

In [28]:
```
train_df = train_df.drop(['AgeBand'], axis=1)
combine = [train_df, test_df]
train_df.head()
```

Out[28]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title
0	0	3	0	1	1	0	7.2500	S	1
1	1	1	1	2	1	0	71.2833	C	3
2	1	3	1	1	0	0	7.9250	S	2
3	1	1	1	2	1	0	53.1000	S	3
4	0	3	0	2	0	0	8.0500	S	1

Create new feature combining existing features

We can create a new feature for FamilySize which combines Parch and SibSp. This will enable us to drop Parch and SibSp from our datasets.

In [29]:
```
for dataset in combine:
    dataset['FamilySize'] = dataset['SibSp'] + dataset['Parch'] + 1

train_df[['FamilySize', 'Survived']].groupby(['FamilySize'],
as_index=False).mean().sort_values(by='Survived', ascending=False)
```

Out[29]:

	FamilySize	Survived
3	4	0.724138
2	3	0.578431
1	2	0.552795
6	7	0.333333
0	1	0.303538
4	5	0.200000
5	6	0.136364
7	8	0.000000
8	11	0.000000

We can create another feature called IsAlone.

In [30]:
```
for dataset in combine:
    dataset['IsAlone'] = 0
    dataset.loc[dataset['FamilySize'] == 1, 'IsAlone'] = 1

train_df[['IsAlone', 'Survived']].groupby(['IsAlone'], as_index=False).mean()
```

Out[30]:

	IsAlone	Survived
0	0	0.505650
1	1	0.303538

Let us drop Parch, SibSp, and FamilySize features in favor of IsAlone.

In [31]:
```
train_df = train_df.drop(['Parch', 'SibSp', 'FamilySize'], axis=1)
test_df = test_df.drop(['Parch', 'SibSp', 'FamilySize'], axis=1)
combine = [train_df, test_df]

train_df.head()
```

```
train_df.head()
```

Out[31]:

	Survived	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone
0	0	3	0	1	7.2500	S	1	0
1	1	1	1	2	71.2833	C	3	0
2	1	3	1	1	7.9250	S	2	1
3	1	1	1	2	53.1000	S	3	0
4	0	3	0	2	8.0500	S	1	1

We can also create an artificial feature combining Pclass and Age.

In [32]:
```
for dataset in combine:
    dataset['Age*Class'] = dataset.Age * dataset.Pclass

train_df.loc[:, ['Age*Class', 'Age', 'Pclass']].head(10)
```

Out[32]:

	Age*Class	Age	Pclass
0	3	1	3
1	2	2	1
2	3	1	3
3	2	2	1
4	6	2	3
5	3	1	3
6	3	3	1
7	0	0	3
8	3	1	3
9	0	0	2

Completing a categorical feature

Embarked feature takes S, Q, C values based on port of embarkation. Our training dataset has two missing values. We simply fill these with the most common occurance.

In [33]:
```
freq_port = train_df.Embarked.dropna().mode()[0]
freq_port
```

Out[33]: 'S'

In [34]:
```
for dataset in combine:
    dataset['Embarked'] = dataset['Embarked'].fillna(freq_port)

train_df[['Embarked', 'Survived']].groupby(['Embarked'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

Out[34]:

	Embarked	Survived
0	C	0.553571
1	Q	0.389610
2	S	0.339009

Converting categorical feature to numeric

We can now convert the EmbarkedFill feature by creating a new numeric Port feature.

In [35]:
```
for dataset in combine:
    dataset['Embarked'] = dataset['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} ).asty
pe(int)

train_df.head()
```

Out[35]:

	Survived	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class
0	0	3	0	1	7.2500	0	1	0	3
1	1	1	1	2	71.2833	1	3	0	2
2	1	3	1	1	7.9250	0	2	1	3
3	1	1	1	2	53.1000	0	3	0	2
4	0	3	0	2	8.0500	0	1	1	6

Quick completing and converting a numeric feature

We can now complete the Fare feature for single missing value in test dataset using mode to get the value that occurs most frequently for this feature. We do this in a single line of code.

Note that we are not creating an intermediate new feature or doing any further analysis for correlation to guess missing feature as we are replacing only a single value. The completion goal achieves desired requirement for model algorithm to operate on non-null values.

We may also want round off the fare to two decimals as it represents currency.

In [36]:
```
test_df['Fare'].fillna(test_df['Fare'].dropna().median(), inplace=True)
test_df.head()
```

Out[36]:

	PassengerId	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class
0	892	3	0	2	7.8292	2	1	1	6
1	893	3	1	2	7.0000	0	3	0	6
2	894	2	0	3	9.6875	2	1	1	6
3	895	3	0	1	8.6625	0	1	1	3
4	896	3	1	1	12.2875	0	3	0	3

We can not create FareBand.

In [37]:
```
train_df['FareBand'] = pd.qcut(train_df['Fare'], 4)
train_df[['FareBand', 'Survived']].groupby(['FareBand'], as_index=False).mean().so
rt_values(by='FareBand', ascending=True)
```

Out[37]:

	FareBand	Survived
0	[0, 7.91]	0.197309
1	(7.91, 14.454]	0.303571
2	(14.454, 31]	0.454955
3	(31, 512.329]	0.581081

Convert the Fare feature to ordinal values based on the FareBand.

In [38]:
```
for dataset in combine:
    dataset.loc[ dataset['Fare'] <= 7.91, 'Fare'] = 0
    dataset.loc[(dataset['Fare'] > 7.91) & (dataset['Fare'] <= 14.454), 'Fare'] =
1
    dataset.loc[(dataset['Fare'] > 14.454) & (dataset['Fare'] <= 31), 'Fare']   =
2
    dataset.loc[ dataset['Fare'] > 31, 'Fare'] = 3
    dataset['Fare'] = dataset['Fare'].astype(int)

train_df = train_df.drop(['FareBand'], axis=1)
combine = [train_df, test_df]

train_df.head(10)
```

Out[38]:

	Survived	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class
0	0	3	0	1	0	0	1	0	3
1	1	1	1	2	3	1	3	0	2
2	1	3	1	1	1	0	2	1	3
3	1	1	1	2	3	0	3	0	2
4	0	3	0	2	1	0	1	1	6
5	0	3	0	1	1	2	1	1	3
6	0	1	0	3	3	0	1	1	3
7	0	3	0	0	2	0	4	0	0
8	1	3	1	1	1	0	3	0	3
9	1	2	1	0	2	1	3	0	0

And the test dataset.

In [39]:
```
test_df.head(10)
```

Out[39]:

	PassengerId	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class
0	892	3	0	2	0	2	1	1	6
1	893	3	1	2	0	0	3	0	6
2	894	2	0	3	1	2	1	1	6
3	895	3	0	1	1	0	1	1	3
4	896	3	1	1	1	0	3	0	3
5	897	3	0	0	1	0	1	1	0
6	898	3	1	1	0	2	2	1	3
7	899	2	0	1	2	0	1	0	2
8	900	3	1	1	0	1	3	1	3
9	901	3	0	1	2	0	1	0	3

Model, predict and solve

Now we are ready to train a model and predict the required solution. There are 60+ predictive modelling algorithms to choose

Now we are ready to train a model and predict the required solution. There are 60+ predictive modelling algorithms to choose from. We must understand the type of problem and solution requirement to narrow down to a select few models which we can evaluate. Our problem is a classification and regression problem. We want to identify relationship between output (Survived or not) with other variables or features (Gender, Age, Port...). We are also perfoming a category of machine learning which is called supervised learning as we are training our model with a given dataset. With these two criteria - Supervised Learning plus Classification and Regression, we can narrow down our choice of models to a few. These include:

- Logistic Regression
- KNN or k-Nearest Neighbors
- Support Vector Machines
- Naive Bayes classifier
- Decision Tree
- Random Forrest
- Perceptron
- Artificial neural network
- RVM or Relevance Vector Machine

In [40]:
```
X_train = train_df.drop("Survived", axis=1)
Y_train = train_df["Survived"]
X_test  = test_df.drop("PassengerId", axis=1).copy()
X_train.shape, Y_train.shape, X_test.shape
```

Out[40]: ((891, 8), (891,), (418, 8))

Logistic Regression is a useful model to run early in the workflow. Logistic regression measures the relationship between the categorical dependent variable (feature) and one or more independent variables (features) by estimating probabilities using a logistic function, which is the cumulative logistic distribution. Reference Wikipedia (https://en.wikipedia.org/wiki/Logistic_regression).

Note the confidence score generated by the model based on our training dataset.

In [41]:
```
# Logistic Regression

logreg = LogisticRegression()
logreg.fit(X_train, Y_train)
Y_pred = logreg.predict(X_test)
acc_log = round(logreg.score(X_train, Y_train) * 100, 2)
acc_log
```

Out[41]: 80.359999999999999

We can use Logistic Regression to validate our assumptions and decisions for feature creating and completing goals. This can be done by calculating the coefficient of the features in the decision function.

Positive coefficients increase the log-odds of the response (and thus increase the probability), and negative coefficients decrease the log-odds of the response (and thus decrease the probability).

- Sex is highest positivie coefficient, implying as the Sex value increases (male: 0 to female: 1), the probability of Survived=1 increases the most.
- Inversely as Pclass increases, probability of Survived=1 decreases the most.
- This way Age*Class is a good artificial feature to model as it has second highest negative correlation with Survived.
- So is Title as second highest positive correlation.

In [42]:
```
coeff_df = pd.DataFrame(train_df.columns.delete(0))
coeff_df.columns = ['Feature']
coeff_df["Correlation"] = pd.Series(logreg.coef_[0])

coeff_df.sort_values(by='Correlation', ascending=False)
```

Out[42]:

	Feature	Correlation

1	Sex	2.201527
5	Title	0.398234
2	Age	0.287163
4	Embarked	0.261762
6	IsAlone	0.129140
3	Fare	-0.085150
7	Age*Class	-0.311200
0	Pclass	-0.749007

Next we model using Support Vector Machines which are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training samples, each marked as belonging to one or the other of **two categories**, an SVM training algorithm builds a model that assigns new test samples to one category or the other, making it a non-probabilistic binary linear classifier. Reference Wikipedia (https://en.wikipedia.org/wiki/Support_vector_machine).

Note that the model generates a confidence score which is higher than Logistics Regression model.

In [43]:
```
# Support Vector Machines

svc = SVC()
svc.fit(X_train, Y_train)
Y_pred = svc.predict(X_test)
acc_svc = round(svc.score(X_train, Y_train) * 100, 2)
acc_svc
```

Out[43]: 83.840000000000003

In pattern recognition, the k-Nearest Neighbors algorithm (or k-NN for short) is a non-parametric method used for classification and regression. A sample is classified by a majority vote of its neighbors, with the sample being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor. Reference Wikipedia (https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm).

KNN confidence score is better than Logistics Regression but worse than SVM.

In [44]:
```
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, Y_train)
Y_pred = knn.predict(X_test)
acc_knn = round(knn.score(X_train, Y_train) * 100, 2)
acc_knn
```

Out[44]: 84.739999999999995

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features) in a learning problem. Reference Wikipedia (https://en.wikipedia.org/wiki/Naive_Bayes_classifier).

The model generated confidence score is the lowest among the models evaluated so far.

In [45]:
```
# Gaussian Naive Bayes

gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
Y_pred = gaussian.predict(X_test)
acc_gaussian = round(gaussian.score(X_train, Y_train) * 100, 2)
```

```
acc_gaussian = round(gaussian.score(X_train, Y_train) * 100, 2)
acc_gaussian
```

Out[45]: 72.280000000000001

The perceptron is an algorithm for supervised learning of binary classifiers (functions that can decide whether an input, represented by a vector of numbers, belongs to some specific class or not). It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The algorithm allows for online learning, in that it processes elements in the training set one at a time. Reference Wikipedia (https://en.wikipedia.org/wiki/Perceptron).

In [46]:
```
# Perceptron

perceptron = Perceptron()
perceptron.fit(X_train, Y_train)
Y_pred = perceptron.predict(X_test)
acc_perceptron = round(perceptron.score(X_train, Y_train) * 100, 2)
acc_perceptron
```

Out[46]: 78.0

In [47]:
```
# Linear SVC

linear_svc = LinearSVC()
linear_svc.fit(X_train, Y_train)
Y_pred = linear_svc.predict(X_test)
acc_linear_svc = round(linear_svc.score(X_train, Y_train) * 100, 2)
acc_linear_svc
```

Out[47]: 79.010000000000005

In [48]:
```
# Stochastic Gradient Descent

sgd = SGDClassifier()
sgd.fit(X_train, Y_train)
Y_pred = sgd.predict(X_test)
acc_sgd = round(sgd.score(X_train, Y_train) * 100, 2)
acc_sgd
```

Out[48]: 77.329999999999998

This model uses a decision tree as a predictive model which maps features (tree branches) to conclusions about the target value (tree leaves). Tree models where the target variable can take a finite set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. Reference Wikipedia (https://en.wikipedia.org/wiki/Decision_tree_learning).

The model confidence score is the highest among models evaluated so far.

In [49]:
```
# Decision Tree

decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)
Y_pred = decision_tree.predict(X_test)
acc_decision_tree = round(decision_tree.score(X_train, Y_train) * 100, 2)
acc_decision_tree
```

Out[49]: 86.760000000000005

The next model Random Forests is one of the most popular. Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees (n_estimators=100) at training time and outputting the class that is the mode of the classes (classification) or mean prediction

(regression) of the individual trees. Reference Wikipedia (https://en.wikipedia.org/wiki/Random_forest).

The model confidence score is the highest among models evaluated so far. We decide to use this model's output (Y_pred) for creating our competition submission of results.

In [50]:
```
# Random Forest

random_forest = RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train, Y_train)
Y_pred = random_forest.predict(X_test)
random_forest.score(X_train, Y_train)
acc_random_forest = round(random_forest.score(X_train, Y_train) * 100, 2)
acc_random_forest
```

Out[50]: 86.760000000000005

Comments (101) All Comments Sort by Hotness

HyesooYoun Preview

Styling with Markdown supported

Enter your comments

Post Comment

Citrus • Posted on Latest Version • 17 days ago • Options • Reply 1

Thank you very much for this tutorial. A very useful introduction for getting started!

ThomasPluck • (2651st in this Competition) • Posted on Latest Version • a month ago • Options • Reply 2

Great tutorial for starting out with essential python libraries.

Manav Sehg... • (2153rd in this Competition) • Posted on Latest Version • 24 days ago • Options • Reply 0

Many thanks for your appreciation. I am glad you find this useful.

Sarine • (3170th in this Competition) • Posted on Latest Version • a month ago • Options • Reply 1

Thank you so much for this tutorial!

Wouter Griffio... • Posted on Latest Version • a month ago • Options • Reply 3

Thank you for this comprehensive tutorial, I find it particularly useful that you put emphasis on the project as a whole including preliminary data analysis. I have a few questions/comments though:

1. I fail to understand your histograms in 'Correlating categorical and numerical features'. We see the number of females/males that survived/did not survive from all three embarkation points, but how do we observe ticket fare from this histogram?
2. I think it is mentioned by Sharan Naribole as well that converting titles into ordinal numbers is not a good idea, for they have no meaning. For instance, you conclude from the Logistic regression that title is a good feature to include, but what is the meaning of this correlation? We cannot say a Mr is better than a Miss who is better than a Master. You should make use of dummies here if you want to include title. I also think dummies might be a good idea for the point of embarkation.
3. Completing a numerical continuous variable, to my opinion, is always a bit arbitrary. You state you prefer method 2 since 1 and 3 introduce random noise. While this is true 1 and 3 on the other hand might lead to less bias than 2 (I have not tested the results though).
4. I noticed a small type when converting age in to bands. You forget to give the highest ages number 4, now you have age categories 0,1,2,3,66,... This also affects Pclass*age (actually you will see the significance of this estimator drops when changed). Next to that it's subject to multi-collinearity as noted by CityofAngels.
5. To fill in some of the ticket fares you mention you use the mode, but I think you're using the median. There's a few other small typos like the one mentioned by Derek Liu.

Keep up the good work!

Manav Sehg... • (2153rd in this Competition) • Posted on Latest Version • 24 days ago • Options • Reply 0

Wow! These are some "well read" comments. You have definitely spent some time on this Kernel and comments. If you have time I am happy to collaborate on a fork here or on GitHub. Let me know if you have time and interest and we can work together to enhance this Notebook for everyone. Post it back on Kaggle when done. What say?

Mike Arango • (2451st in this Competition) • Posted on Latest Version • a month ago • Options • Reply 1

Thanks for sharing this notebook! It was a great intro to kaggle and data science projects. In describing a histogram right before you plot the histogram for age you write: "Note that x-axis in historgram visualizations represents the count of samples or passengers." I think you meant to say y-axis as this is usually where frequency/count is displayed.

Vamsee Ragi • (736th in this Competition) • Posted on Latest Version • a month ago • Options • Reply

1

Thanks for your efforts and sharing with us. It is extremely helpful for the new comers. I appreciate that

Manav Sehg... • (2153rd in this Competition) • Posted on Latest Version • a month ago • Options • Reply

0

You are very welcome Vamsee. I hope to continue creating more such kernels for folks new to Kaggle and data science.

James Thomps... • Posted on Latest Version • 2 months ago • Options • Reply

1

Very helpful thankyou :)

byt3 • (1431st in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply

1

Thank you for putting so much time and effort into this great tutorial.

Manav Sehg... • (2153rd in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply

1

You are very welcome. I really enjoyed doing so and if it helps the Kaggle new joiners, like me, then my goal is achieved.

O. Yang • Posted on Latest Version • 2 months ago • Options • Reply

1

A good post to start with. I found the part of 'Correlating categorical and numerical features' is especially helpful. Thanks.

JineshShah • Posted on Latest Version • 2 months ago • Options • Reply

1

Thanks for sharing. Quite Insightful. I have a doubt regarding conversion of categorical variables to numerical. Doesn't the current form imply ordering of categorical variables like Embarked (3 > 2 > 1) ? When should one do one-hot encoding for the same ? Thanks.

Schmitzi • (1033rd in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply

2

Thanks for sharing this very well written Kernel. I am quite new to scikit-learn so this is an excellent example for testing different methods an algorithms.

Manav Sehg... • (2153rd in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply ^ 0

Many thanks for your appreciation.

Nat • (5267th in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply ^ 1 v

Thanks for sharing your work, it helped me a lot :)

Manav Sehg... • (2153rd in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply ^ 0

@Nat glad to know you found this work useful for yours.

lillee • Posted on Latest Version • 2 months ago • Options • Reply ^ -1 v

guys help, where can i find a dataset for packet inspection systems

Vamsee Ragi • (736th in this Competition) • Posted on Latest Version • a month ago • Options • Reply ^ 0 v

Hi, i don't see any dataset with the name packet inspection systems in the dataset search. Sorry, if my answer is not correct or not related to your question. I am new to Kaggle

lillee • Posted on Latest Version • 2 months ago • Options • Reply ^ -1 v

lillee • Posted on Latest Version • 2 months ago • Options • Reply ^ -1 v

guys help, where can i find a dataset for packet inspection systems

SergioBadillo • (6324th in this Competition) • Posted on Latest Version • 3 months ago • Options • Reply ^ 1 v

Hello Startupsci, thanks for this well explained notebook!

I am using following step by step to learn terms and practice the methodology. I have a question. In your subsection "Assumptions based on data analysis", you include the original three assumptions of the description under a "classifying" subsubtitle:

> Women (Sex=female) were more likely to have survived. Children (Age?) were more likely to have survived. The upper-class passengers (Pclass=1) were more likely to have survived.

I wonder why these are not mentioned just at the "Correlating" subsubtitle at the very top of the section? Aren't those correlation related tasks that might be tested/included/discarded at early stages? I'm sorry for the dumbness, I just wanna get the concepts right and that created a small distraction.

Have a nice day!

Manav Sehg... • (2153rd in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply 0

Good observation @SergioBadillo and you are right, these points could be part of correlation section as well. I wanted to highlight few examples in each section as I am demonstrating how one can use the 7Cs method to think through structuring the solution activities. I am using classifying and category creation interchangeably here. This may cause some confusion which I may address in a later update to this notebook. For now read classifying tasks as "dealing with categorical data" - so age bands, or F/M sex, or Pclass are all categorical data.

Reinhard • Posted on Version 5 • 4 months ago • Options • Reply 5

It's a very well written kernel. I especially like the histograms showing the correlation between the survival count, age, and class.

May I add some remarks on your analysis: I don't know how you found that 35% of the passengers had siblings aboard. The SibSp feature describes the number of spouses plus the number of siblings. I did some analysis on the relations of passengers and found that about 17% of the passengers had spouses and 16% had siblings aboard. I think the duplicate ticket ids are no error. All passengers with the same ticket id have the same fare value as well. I assume that tickets were valid for several passengers. This would also explain why many passengers with the same surname had the same ticket id and why there is a correlation between the number of people with the same ticket and the fare. For this reason it might be better to divide the fare value by the number of people with the same ticket. I doubt that there is a real correlation between the place of embarkment and the survival rate. I think the reason why the survival rate of passengers who embarked at C is higher than at Q is that the majority of first class passengers boarded at C whereas almost only third class passengers boarded at Q. When you analysed the logistic regression model you compared the regression slopes and not the regression coefficients with each other (regression coefficients are between -1 and 1). Negative correlation coefficients do not mean that the importance of the feature they are associated with is low, it means that a high value of the feature results in a low value of the output, e.g. Pclass should have a negative correlation coefficient as 3rd class passengers were less likely to survive than 1st class passengers.

Manav Sehg... • (2153rd in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply 0

@Reinhard - many thanks for your suggestions. I have fixed and updated the notebook with acknowledgements.

Sharan Naribol... • Posted on Version 5 • 4 months ago • Options • Reply 4

Insightful workflow! Thanks for sharing. I would perform a Gridsearch over different parameters of each prediction model for future submissions.

EDIT: Few comments:

1) Although the Cabin feature has missing entries, it would be interesting to interpolate data from Ticket and Pclass/Fare to complete the Cabin training and test data.

2) It might be worth the effort to convert categorical features to numerical using pandas get_dummies() function. In this case, Embarked, Title converted to integers does not really have significance. If numbers are a rating where 1 is better than 2, 2 is better than 3 and so forth it may make sense to use them. If the numbers relate to a condition like color (e.g. 1 is blue, 2 is green, 3 is red, etc.), it would be better to convert to dummies (giving consideration to degrees of freedom).

Sharan Naribol... • Posted on Version 5 • 4 months ago • Options • Reply 2

@Startupsci I found an error in the way you are converting Categorical to Numerical variables. In your code,

Titles = list(enumerate(np.unique(train_df['Title']))) Titles_dict = { name : i for i, name in Titles }
train_df['TitleBand'] = train_df.Title.map(lambda x: Titles_dict[x]).astype(int)

Titles = list(enumerate(np.unique(test_df['Title'])))

In this case, the Training set and Testing set will have different Numerical values for same Title. In this case, Mr. is assigned 12 in training set and Mr. is assigned 5 in testing set.

Manav Sehg... • (2153rd in this Competition) • Posted on Version 10 • 3 months ago • Options • Reply 0

@Sharan Naribole, many thanks for pointing this out. I have fixed this issue in 2017-01-29 release of this notebook.

esigler • (3687th in this Competition) • Posted on Latest Version • a month ago • Options • Reply 0

Thank you for sharing!

Sharan Naribol... • Posted on Version 5 • 4 months ago • Options • Reply 0

1. The Fare per person is found to vary significantly not only as function of Pclass, Embarked, gender but also Cabin Deck (first letter of Cabin). I tried using the shared Ticket numbers for filling missing Cabin values but was able to fill only 11 values.
2. For fitting a fare for children, unfortunately, most children are accompanied by parents or siblings. If we try to get true fare for a child, there is a single passenger who is below 10 years and with SibSp =0 and Parch=0. The Passenger (ID=778) who is female and 5 years old and Fare = 12.475. EDIT: There are actually 16 more such passengers in test data.
3. Interestingly, in Pclass =2 and Pclass = 3, the Fare Per Person for ages > 60 is among the lowest in their respective classes.

Kofi • Posted on Latest Version • 3 months ago • Options • Reply 0

Hey, I've been trying to accomplish what you did for part 1 there with no success. I really just want to do it for coding practice purposes. Any help would be greatly appreciated.

I've tried this approach, to group the ticket cabins and get the first of each, skipping the nulls and storing them in 'Cabin' but it gives me an error "TypeError: cannot use label indexing with a null key".

df['Cabin'] = df.groupby(['Ticket'])['Cabin'].apply(lambda x: x.loc[x.first_valid_index()]).

Ive also tried just creating a dataframe (code below), with the tickets that will give me cabin values for duplicates and tried merging it into the training df with no success either.

df2 = df[(df.duplicated('Ticket')) & (df['Cabin'].notnull())][['Ticket','Cabin']].drop_duplicates('Ticket')

Kofi • Posted on Latest Version • 3 months ago • Options • Reply 0

Nvm I thought of a solution; ticket_dupl = maindf.groupby('Ticket')['Cabin'].first().reset_index() then merge with the maindf on ticket, imputes just 11

Sharan Narib... • Posted on Latest Version • 2 months ago • Options • Reply 0

Okay good to hear you figured it out!

EY • (198th in this Competition) • Posted on Latest Version • a month ago • Options • Reply 0

Thank you and really nice explanations for understanding how to do data analysis and finally modeling.

Alexander Lyab... • Posted on Version 5 • 4 months ago • Options • Reply

0

```
test_df['Sex'].map( {'female': 1, 'male': 0} ).astype(int)
```

Do you need astype(int) here??

Alexander Lyab... • Posted on Version 5 • 4 months ago • Options • Reply

0

I have a question about Age prediction.

We have around 20% of lost age in both datasets. Even though the prediction age is pretty strain forward - don't you think it would be better to unite train and test data to have a bigger dataset from prediction?

This is a difference between current prediction and combinated prediction

```
array([[-2. ,  0.5,  0. ],
       [-1. ,  0. , -0.5]])
```

Why can't we use sklearn for age prediction in the same way as we leant Surviving?

Thank you

prodofchem • Posted on Latest Version • 2 months ago • Options • Reply

0

You can use LinearRegression to predict the missing ages. You could get a few nonsensical values, like negative ages, so watch out for those.

Alexander Lyab... • Posted on Version 5 • 4 months ago • Options • Reply

0

I also find an issue with idea of using Title.

train_df doesn't have title 'Dona.' but test_df has.

11245 • (6804th in this Competition) • Posted on Latest Version • 7 days ago • Options • Reply

0

Nice job! Thank you!

JineshShah • Posted on Latest Version • 2 months ago • Options • Reply

0

Ibah • (495th in this Competition) • Posted on Version 5 • 4 months ago • Options • Reply

0

@Reinhard Following up on your suggestion on dividing the fare value by the number of people with the same ticket (= 'fare per person').

Comparing distributions of fare value conditional on the class, shows there's much overlap between the distributions for different classes. It suggests that many 2nd class and especially 3rd class passengers paid for their tickets as much as if travelling 1st class, which looks suspicious.

Repeating the same comparison or the fare value divided by the number of people with the same ticket ('fare per person') shows the distributions less spread and neatly separated. This suggests that one should rather use the 'fare per person' than the simple fare value as it depends on the number of people buying a ticket together. Still maybe some discounts for groups were available making the 'fare per person' underestimating the value of the tickets shared by a group of passengers.

Unknown • (6230th in this Competition) • Posted on Latest Version • 12 days ago • Options • Reply

0

Thanks for this tutorial. Solved my first ML problem.

NiranjanRame... • Posted on Latest Version • 4 days ago • Options • Reply

0

Hello Sir, I have a pretty newbie doubt. So you take the mean of Survived value grouping by a particular feature say Pclass, we get values like : Pclass 1: Survived=0.629630, Pclass 2: Survived= 0.472826 and Pclass 3: Survived= 0.242363 Are these values the probability of people surviving when they belong to the respective class?

Ajinkya Korde • Posted on Latest Version • 2 months ago • Options • Reply

0

Thanks for sharing, very helpful!

ZhiboYang • (2270th in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply

0

In the logistic regression part, why you use 'log.coef_' to determine whether a estimator is good? Why don't you use correlation matrix instead? Thanks

mumumuyany... • Posted on Version 10 • 3 months ago • Options • Reply

0

it enlightens me, thanks

Reinhard • Posted on Version 5 • 4 months ago • Options • Reply

0

@Ibah, another way to check that the fare depends on the number of passengers with the same ticket is to make a linear regression of fare and fare per person with respect to the number of passengers with the same ticket. In case of the pure fare values is a clear increase with the number of passengers whereas in case of fare per person the slope of the regression line is much smaller.

Apart from discounts, another reason why bigger groups had to pay less fare per person could be that these groups included children whose fare was lower than the fare of adults. Could be interesting to fit a different fare value for passengers below a certain age.

Alex • Posted on Latest Version • 22 days ago • Options • Reply

0

Awesome tutorial, thanks for taking the time :)

jiahui604 • (5811th in this Competition) • Posted on Latest Version • a month ago • Options • Reply

0

Thanks for sharing this great code! I tried to run this in Pycharm but got a track back from the following code, it said "AttributeError: 'module' object has no attribute 'hist' ", I'm wondering how can I solve this problem, and why this happened? Thanks again!

```
grid = sns.FacetGrid(train_df, row='Pclass', col='Sex', size=2.2, aspect=1.6)
grid.map(plt.hist, 'Age', alpha=.5, bins=20)
grid.add_legend()
```

O. Yang • Posted on Latest Version • 2 months ago • Options • Reply

0

A good post to start with. I found the part of 'Correlating categorical and numerical features' is especially helpful. Thanks.

rlchen • Posted on Latest Version • 6 days ago • Options • Reply

0

Thank you for your tutorial.it's very useful to me.

Manav Sehgal • (2153rd in this Competition) • Posted on Version 5 • 4 months ago • Options • Reply

0

@Reinhard Thank you for your insightful and detailed comments. These do make sense to me and I will make corrections to the kernel accordingly.

Manav Sehgal • (2153rd in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply

0

Thank you everyone for your comments and vote of thanks!

Manav Sehgal • (2153rd in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply 0

Please continue to vote this notebook so others can find it.

Manav Sehg... • (2153rd in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply 0

Let us target 100 votes please.

Manav Sehgal • (2153rd in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply 0

Please share this notebook with others who might find it useful.

Manav Sehgal • (2153rd in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply 0

This notebook is part of new data science solutions book available from Amazon in print and ebook https://startupsci.com

ShaharBarak • Posted on Version 5 • 4 months ago • Options • Reply 0

Thank you for the elegant kernel!

Chen Shen • Posted on Latest Version • 2 months ago • Options • Reply 0

Really helpful ! Thanks :)

Mike Arango • (2451st in this Competition) • Posted on Latest Version • a month ago • Options • Reply 0

Mike Arango • (2451st in this Competition) • Posted on Latest Version • a month ago • Options • Reply 0

When you use the 'Age Band' variable to turn 'Age' into an ordinal variable you write: `for dataset in combine: dataset.loc[ dataset['Age'] <= 16, 'Age'] = 0 dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 32), 'Age'] = 1 dataset.loc[(dataset['Age'] > 32) & (dataset['Age'] <=`

```
48), 'Age'] = 2 dataset.loc[(dataset['Age'] > 48) & (dataset['Age'] <= 64), 'Age'] = 3
dataset.loc[ dataset['Age'] > 64, 'Age'] train_df.head()
```

but don't you still have to specify: `dataset.loc[ dataset['Age'] > 64, 'Age'] = 4` ?

I could be wrong, but I don't think it assigns those values to 4.

Bishal Lakha • (3244th in this Competition) • Posted on Latest Version • a month ago • Options • Reply

0

Thank you very much! This was very helpful.

zzzsss • Posted on Latest Version • 10 days ago • Options • Reply

0

Very helpful !! thank you very much!!

Max • (608th in this Competition) • Posted on Latest Version • 5 days ago • Options • Reply

0

Thank you for this tutorial! It is a great intro for newcomers.

Matan • Posted on Latest Version • 3 months ago • Options • Reply

0

Matan • Posted on Latest Version • 3 months ago • Options • Reply

0

someone can explain this code: guess_ages[i,j] = int(age_guess/0.5 + 0.5) * 0.5

tnx :)

Glenn Reyno... • Posted on Latest Version • 25 days ago • Options • Reply

0

Rounding to nearest 0.5?

Warheads • Posted on Latest Version • 2 months ago • Options • Reply

0

wonderful .

Jiyang Li • (101st in this Competition) • Posted on Latest Version • 11 days ago • Options • Reply

0

thank you for sharing!!

Weikai • (1302nd in this Competition) • Posted on Latest Version • 16 days ago • Options • Reply 0

Very helpful, especially for beginners!

Weikai • (1302nd in this Competition) • Posted on Latest Version • 16 days ago • Options • Reply 0

Weikai • (1302nd in this Competition) • Posted on Latest Version • 16 days ago • Options • Reply 0

Weikai • (1302nd in this Competition) • Posted on Latest Version • 16 days ago • Options • Reply 0

Weikai • (1302nd in this Competition) • Posted on Latest Version • 16 days ago • Options • Reply 0

CityofAngels • (4102nd in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply 0

Just a few quick comments:

Aggressive age-banding isn't reallllllly necessary for random-forest and in fact the model performs significantly better without it any age bands. (I got to a score of 95 with slight tweaking and random forest). It does improve the score of the average model though.

Having Age*Class in there along with Age and PClass is excessive collinearity for sure. Even if it doesn't change the scores, there's no way that's proper procedure. If you want to combine the two into one variable, I would change the banding of the variable that starts as 0 to start at 1. Multiplying by 0 removes the linear correlation between the other variable and Age*Class.

Also, XGBoost by itself will get you to 96+ with minor tweaking and in my humble opinion should almost always be considered in any algorithm comparison.

Manav Sehg... • (2153rd in this Competition) • Posted on Latest Version • 2 months ago • Options • Reply 0

Thanks @CityofAngles. You make some interesting observations. I specially like the collinearity issue you point out. Of course I will keep the notebook as is for now so that it serves "also" as an example of what NOT to do in some cases during a competition solution workflow - hopefully

readers follow your comment along with the notebook to get the complete picture. If you have a fork of this notebook with XGBoost, I will really appreciate a link here and back. I am adding XGBoost in my future work where it makes sense.

v歪歪vv • Posted on Latest Version • a month ago • Options • Reply 0

Thanks very much.

Derek Liu • (5342nd in this Competition) • Posted on Latest Version • a month ago • Options • Reply 0

“We can not remove the AgeBand feature.” Dose this sentence mean "We can **now** remove the AgeBand feature."

Fantastic Ho... • (3709th in this Competition) • Posted on Latest Version • a month ago • Options • Reply 0

I think so. I delete "AgeBand" column.

Michael Colem... • (3694th in this Competition) • Posted on Latest Version • a month ago • Options • Reply 0

This is very helpful!

I spotted a few typographical errors if you'd like to have them.

Kevin Ma • (571st in this Competition) • Posted on Latest Version • a month ago • Options • Reply 0

Is this a bug? Line 41

```
logreg.fit(X_train, Y_train)
acc_log = round(logreg.score(X_train, Y_train) * 100, 2)
```

You train and eval the model on the same set of data?

Glenn Reyno... • Posted on Latest Version • 25 days ago • Options • Reply 0

The test dataset has no ground truth, so you need to check the degree of fit to your model on the training dataset. Then you run the best one on the test dataset and submit results.

Glenn Reynolds • Posted on Latest Version • 25 days ago • Options • Reply

0

Very helpful introduction to Kaggle.

BearKai • Posted on Latest Version • 23 days ago • Options • Reply

0

This notebook introduces a typical workflow in detail. It helps me (a greenhand) a lot! Thanks for your sharing!

nealzGu • Posted on Latest Version • 24 days ago • Options • Reply

0

Great work! Loved it!

First question: [in 44] *knn = KNeighborsClassifier(n_neighbors = 3)*

why 3 neighbors? don't we just need 2 (survived/ not survived) ??

Thanks!

BearKai • Posted on Latest Version • 23 days ago • Options • Reply

1

You may misunderstand the KNN classifier. The number of nearest neighbors has no direct relationship to the number of classes. For example, we set k=5 (k is usually an odd number), and one neighbor belongs to class A, one belongs to class B, three belongs to C, then we assign the test sample to class C. Note that there may be class D, class E, i.e., zero neighbor belongs to C/D, we do not care, since 3 is the biggest.

tang xuan • (3850th in this Competition) • Posted on Latest Version • 17 days ago • Options • Reply

0

Thank you.

bigheart • (6167th in this Competition) • Posted on Latest Version • 24 days ago • Options • Reply

0

nice notebook! thx~

leonjay2012 • (5029th in this Competition) • Posted on Latest Version • 9 days ago • Options • Reply

0

A brilliant textbook solution for us newcomers! Appreciate

Mark Ma • (2929th in this Competition) • Posted on Latest Version • 2 days ago • Options • Reply

0

As a new comer, I've learned lots of valuable and practical skills in ML from this tutorial. Thanks!

Tianjun Ma • (6941st in this Competition) • Posted on Latest Version • 3 days ago • Options • Reply

0

Nice tutorial.

Tianjun Ma • (6941st in this Competition) • Posted on Latest Version • 3 days ago • Options • Reply

0

Ray Xu • Posted on Latest Version • 3 days ago • Options • Reply

0

Just read and try carefully. Thanks a lot.

Michele Pariani • (5426th in this Competition) • Posted on Latest Version • 19 days ago • Options • Reply

0

Thank you for guiding us through preprocessing!

dBabbar • Posted on Latest Version • 14 days ago • Options • Reply

0

Thanks for the notebook I have done the Andrew NG course on coursera after that am starting kaggle. although i studied ML in octave but i was still able to go through your notebook as i know basic python. can you suggest me a good way for studying further, should i do a fast course on ML in python on just move on with kaggle.

Manav Sehg... • (2153rd in this Competition) • Posted on Latest Version • 10 days ago • Options • Reply

0

Congrats on completing the Andrew NG course. It is a really good start. For Python + Data Science check out https://www.datacamp.com/ videos.

running_kai • (2804th in this Competition) • Posted on Latest Version • 14 days ago • Options • Reply

0

Thank you very much! Very good tutorial!

Manav Sehg... • (2153rd in this Competition) • Posted on Latest Version • 10 days ago • Options • Reply

1

Many thanks! Please vote and share if you like (smiles)

Timothé Didier... • (4324th in this Competition) • Posted on Latest Version • 11 days ago • Options • Reply

0

Hi, Thank you for this good tutorial. I am new here and it was a perfect and very useful introduction for getting started with Kaggle competitions. I would like to ask you one question : **In this example we tried both logistic regression model and linear SVM model. I thought these two models were actually doing the same thing. What is the difference?** Thanks in advance

Trung Nguyen ... • Posted on Latest Version • 10 days ago • Options • Reply

0

Thank you so much for this tutorial