Project: Titanic - Machine Learning from Disaster

Overview

1. Exploring the data
2. Data Cleaning / Preprocessing
3. Model Building
4. Results

```
md"""
# Project: Titanic - Machine Learning from Disaster

## Overview
1) Exploring the data
2) Data Cleaning / Preprocessing
3) Model Building
4) Results
"""
```

```
using Plots
```

```
using CSV
```

```
using DataFrames
```

```
using ScikitLearn
```

```
using VegaLite
```

```
using Statistics
```

```
using Distributions
```

	PassengerId	Survived	Pclass	Name	Sex	
1	1	0	3	"Braund, Mr. Owen Harris"	"male"	22.
2	2	1	1	"Cumings, Mrs. John Bradley (Florence	"female"	38.
3	3	1	3	"Heikkinen, Miss. Laina"	"female"	26.
4	4	1	1	"Futrelle, Mrs. Jacques Heath (Lily Ma	"female"	35.
5	5	0	3	"Allen, Mr. William Henry"	"male"	35.
6	6	0	3	"Moran, Mr. James"	"male"	mis
7	7	0	1	"McCarthy, Mr. Timothy J"	"male"	54.
8	8	0	3	"Palsson, Master. Gosta Leonard"	"male"	2.0
9	9	1	3	"Johnson, Mrs. Oscar W (Elisabeth Vilh	"female"	27.
10	10	1	2	"Nasser, Mrs. Nicholas (Adele Achem)"	"female"	14.
more						
891	891	0	3	"Dooley, Mr. Patrick"	"male"	32.

```julia
begin
    # load Train Data
    path_train = "train.csv"
    data_train = CSV.read(path_train, DataFrame)
end
```

	PassengerId	Pclass	Name	Sex	Age	Sil
1	892	3	"Kelly, Mr. James"	"male"	34.5	0
2	893	3	"Wilkes, Mrs. James (Ellen Needs)"	"female"	47.0	1
3	894	2	"Myles, Mr. Thomas Francis"	"male"	62.0	0
4	895	3	"Wirz, Mr. Albert"	"male"	27.0	0
5	896	3	"Hirvonen, Mrs. Alexander (Helga E Lin	"female"	22.0	1
6	897	3	"Svensson, Mr. Johan Cervin"	"male"	14.0	0
7	898	3	"Connolly, Miss. Kate"	"female"	30.0	0
8	899	2	"Caldwell, Mr. Albert Francis"	"male"	26.0	1
9	900	3	"Abrahim, Mrs. Joseph (Sophie Halaut E	"female"	18.0	0
10	901	3	"Davies, Mr. John Samuel"	"male"	21.0	2
more						
418	1309	3	"Peter, Master. Michael J"	"male"	missing	1

```
begin
    # load Test Data
        path_test = "test.csv"
        data_test = CSV.read(path_test, DataFrame)
end
```

Exploring the data

	variable	mean	min	median	max
1	:PassengerId	446.0	1	446.0	891
2	:Survived	0.383838	0	0.0	1
3	:Pclass	2.30864	1	3.0	3
4	:Name	nothing	"Abbing, Mr. Anthony"	nothing	"van Melkebeke, Mr. Philemo
5	:Sex	nothing	"female"	nothing	"male"
6	:Age	29.6991	0.42	28.0	80.0
7	:SibSp	0.523008	0	0.0	8
8	:Parch	0.381594	0	0.0	6
9	:Ticket	nothing	"110152"	nothing	"WE/P 5735"
10	:Fare	32.2042	0.0	14.4542	512.329
11	:Cabin	nothing	"A10"	nothing	"T"
12	:Embarked	nothing	"C"	nothing	"S"

```
describe(data_train)
```

	variable	mean	min	median	m
1	:PassengerId	1100.5	892	1100.5	1309
2	:Pclass	2.26555	1	3.0	3
3	:Name	nothing	"Abbott, Master. Eugene Joseph"	nothing	"van Billiard, Ma
4	:Sex	nothing	"female"	nothing	"male"
5	:Age	30.2726	0.17	27.0	76.0
6	:SibSp	0.447368	0	0.0	8
7	:Parch	0.392344	0	0.0	9
8	:Ticket	nothing	"110469"	nothing	"W.E.P. 5734"
9	:Fare	35.6272	0.0	14.4542	512.329
10	:Cabin	nothing	"A11"	nothing	"G6"
11	:Embarked	nothing	"C"	nothing	"S"

```
describe(data_test)
```

Data Overview / Plots

Numeric Data

- Age
- SibSp (Siplings and Spouses)
- Parch (Partens and Children)
- Fare

Plots for numeric data

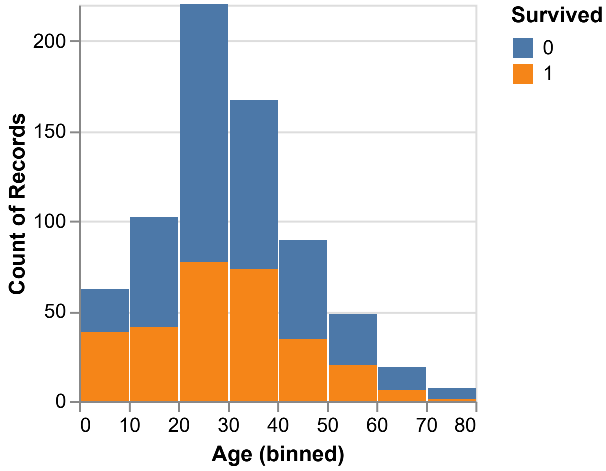
- Histograms to understand distributions
- Correlation Plot

numeric_df =

	Age	SibSp	Parch	Fare
1	22.0	1	0	7.25
2	38.0	1	0	71.2833
3	26.0	0	0	7.925
4	35.0	1	0	53.1
5	35.0	0	0	8.05
6	missing	0	0	8.4583
7	54.0	0	0	51.8625
8	2.0	3	1	21.075
9	27.0	0	2	11.1333
10	14.0	1	0	30.0708
more				
891	32.0	0	0	7.75

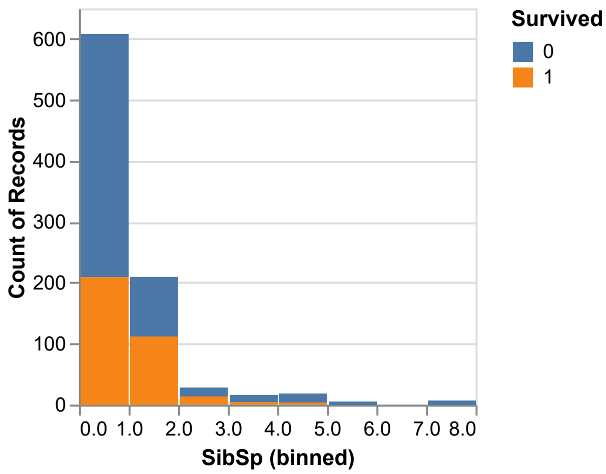
```
numeric_df = data_train[:,[:Age,:SibSp,:Parch,:Fare]]
```

hist_Age =



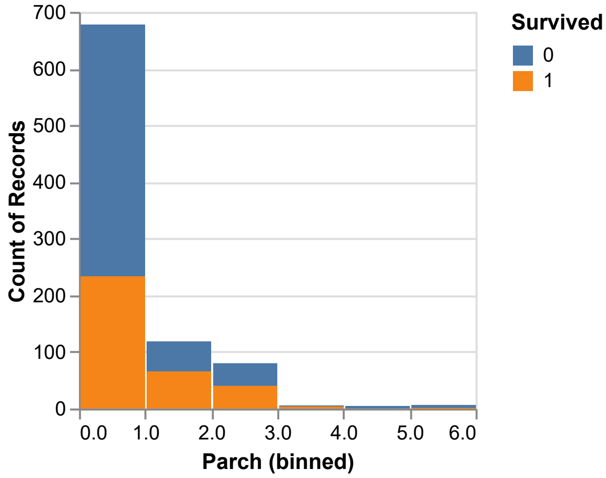

```
hist_Age= @vlplot(data=data_train)+
    @vlplot(:bar, x={:Age, bin=true}, y="count()", color={:Survived, type =
    "nominal"})
```

hist_SibSp =



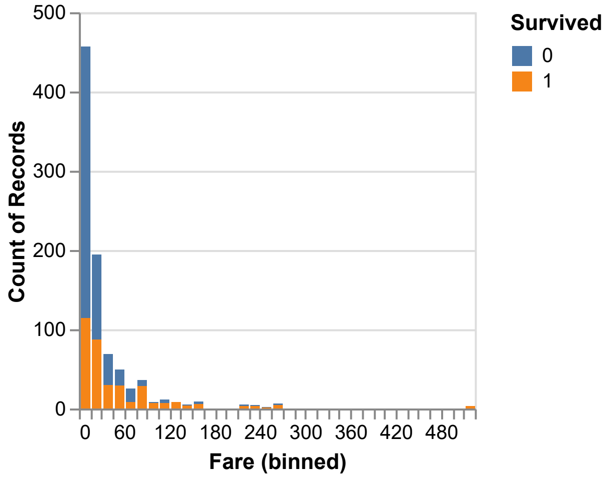

```
hist_SibSp= @vlplot(data=data_train)+
    @vlplot(:bar, x={:SibSp, bin=true}, y="count()", color={:Survived, type =
    "nominal"})
```

hist_Parch =




```
hist_Parch = @vlplot(data=data_train)+
    @vlplot(:bar, x={:Parch , bin=true}, y="count()", color={:Survived, type =
    "nominal"})
```

hist_Fare =



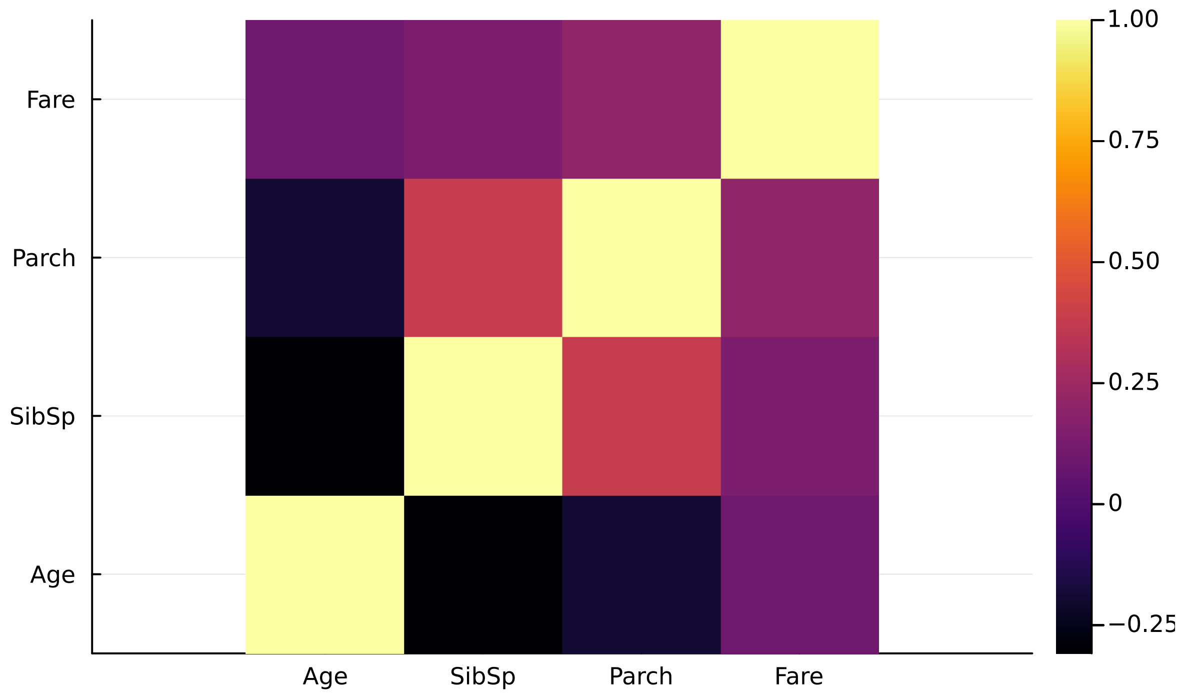

```
hist_Fare = @vlplot(data=data_train)+
    @vlplot(:bar, x={:Fare, bin={step=15}}, y="count()", color={:Survived, type =
    "nominal"})
```

```
cor_numeric = 4×4 Matrix{Float64}:
                 1.0        -0.308247  -0.189119  0.0960667
                -0.308247    1.0        0.38382   0.138329
                -0.189119    0.38382    1.0       0.205119
                 0.0960667   0.138329   0.205119  1.0
```

```
cor_numeric=cor(Matrix(dropmissing(numeric_df)))
```

```
GRBackend()
```

```
gr()
```




```
Plots.heatmap(names(numeric_df),names(numeric_df),cor_numeric,aspect_ratio = 1)
```

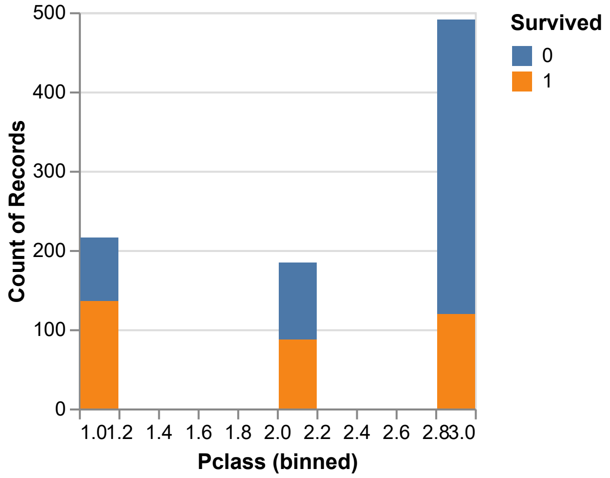
Categorical

- Survived
- Pclass
- Sex
- *(Cabin)*
- Embarked

Plots for Categorical Data

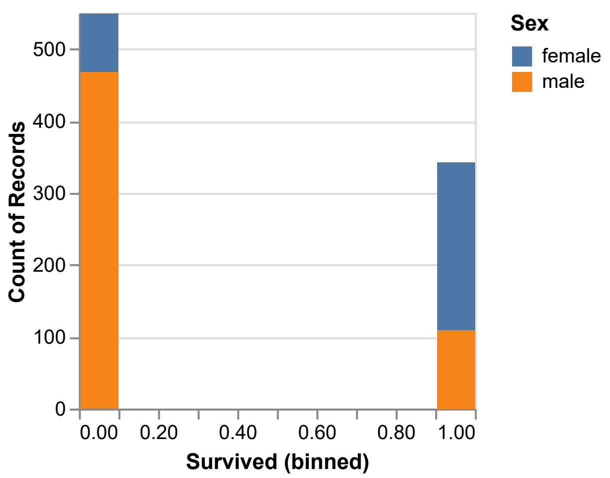
- Bar charts to understand balance of classes

bar_Pclass =



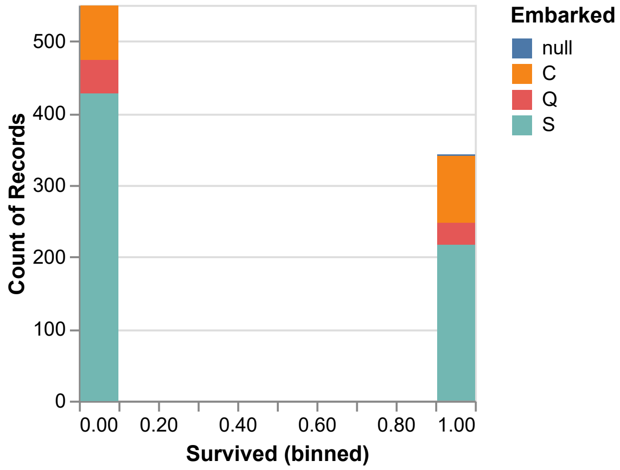

```
bar_Pclass = @vlplot(data=data_train)+
    @vlplot(:bar, x={:Pclass , bin=true}, y="count()", color={:Survived, type =
    "nominal"})
```

bar_Sex =




```
bar_Sex = @vlplot(data=data_train)+
    @vlplot(:bar, x={:Survived , bin=true}, y="count()", color={:Sex, type =
    "nominal"})
```

bar_Embarked =




```
bar_Embarked = @vlplot(data=data_train)+
    @vlplot(:bar, x={:Survived , bin=true}, y="count()", color={:Embarked, type =
    "nominal"})
```

Other Data

- Name
- Ticket

```
891-element SentinelArrays.ChainedVector{String, Vector{String}}:
 "Braund, Mr. Owen Harris"
 "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
 "Heikkinen, Miss. Laina"
 "Futrelle, Mrs. Jacques Heath (Lily May Peel)"
 "Allen, Mr. William Henry"
 "Moran, Mr. James"
 "McCarthy, Mr. Timothy J"
 ⋮
 "Rice, Mrs. William (Margaret Norton)"
 "Montvila, Rev. Juozas"
 "Graham, Miss. Margaret Edith"
 "Johnston, Miss. Catherine Helen \"Carrie\""
 "Behr, Mr. Karl Howell"
 "Dooley, Mr. Patrick"
```

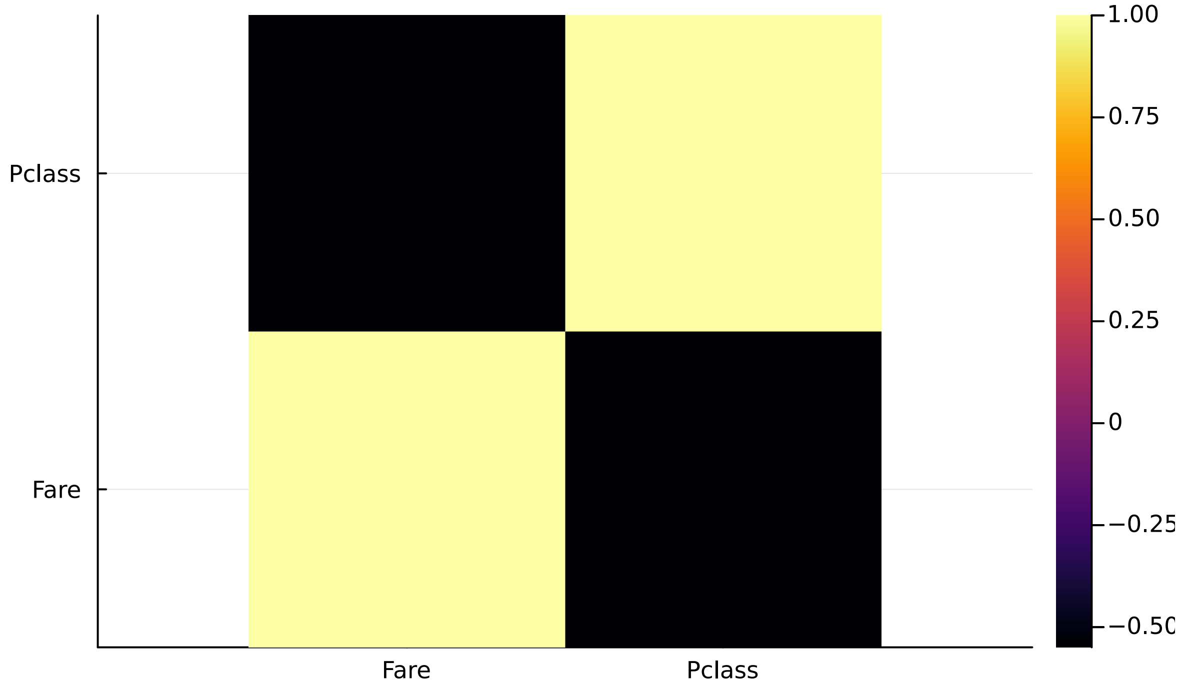
```
data_train.Name
```

```
891-element SentinelArrays.ChainedVector{String31, Vector{String31}}:
 "A/5 21171"
 "PC 17599"
 "STON/O2. 3101282"
 "113803"
 "373450"
 "330877"
 "17463"
 ⋮
 "382652"
 "211536"
 "112053"
 "W./C. 6607"
 "111369"
 "370376"
```

```
data_train.Ticket
```

```
2×2 Matrix{Float64}:
  1.0      -0.5495
 -0.5495    1.0
```

```
begin
    # correlation between Pclass and Fare
    dat = dropmissing(select(data_train, [:Fare, :Pclass]))
    cor_Pclass_Fare=cor(Matrix(dat))
end
```




```
begin
    gr()
    Plots.heatmap(names(dat),names(dat),cor_Pclass_Fare,aspect_ratio = 1)
end
```

Conclusion

Dealing with missing values

The training data set has a total of 891 samples.

- Age
  - 177 missing values (in training data)
  - replace missing values with mean $\pm$ sd
- Cabin
  - 687 missing values (in training data)
  - don't use Cabin as feature
- Embarked
  - 2 missing values (in training data)
  - drop missing
- Fare
  - high (negative) correlation with class
  - if missing: replace with mean for the corresponding class

Feature Selection

Y = Survived

- Exclude:
  - PassengerId
  - Name
  - Ticket
  - Cabin
- Include:
  - Pclass
  - Sex
  - Age
  - SibSp and Parch
  - Fare
  - Embarked

Data Cleaning

```
get_test_data (generic function with 1 method)
```

```
include("project_functions.jl")
```

```
PyObject <class 'sklearn.preprocessing._data.StandardScaler'>
```

```
@sk_import preprocessing: StandardScaler
```

```
(889×11 Matrix{Float64}:                                                              , [0, 1, 1, 1, 0, 0, 0,
  22.0     0.1  1.0  0.0  0.0  1.0  0.0  0.0141511  1.0  0.0  0.0
  38.0     0.1  0.0  1.0  0.0  0.0  1.0  0.139136   0.0  1.0  0.0
  26.0     0.0  1.0  0.0  0.0  0.0  1.0  0.0154686  1.0  0.0  0.0
  35.0     0.1  0.0  1.0  0.0  0.0  1.0  0.103644   1.0  0.0  0.0
  35.0     0.0  1.0  0.0  0.0  1.0  0.0  0.0157126  1.0  0.0  0.0
  27.1101  0.0  1.0  0.0  0.0  1.0  0.0  0.0165095  0.0  0.0  1.0
  54.0     0.0  0.0  1.0  0.0  1.0  0.0  0.101229   1.0  0.0  0.0
   ⋮                            ⋮                             ⋮
  39.0     0.5  1.0  0.0  0.0  0.0  1.0  0.0568482  0.0  0.0  1.0
  27.0     0.0  0.0  0.0  1.0  1.0  0.0  0.0253743  1.0  0.0  0.0
  19.0     0.0  0.0  1.0  0.0  0.0  1.0  0.0585561  1.0  0.0  0.0
  14.1029  0.3  1.0  0.0  0.0  0.0  1.0  0.0457714  1.0  0.0  0.0
  26.0     0.0  0.0  1.0  0.0  1.0  0.0  0.0585561  0.0  1.0  0.0
  32.0     0.0  1.0  0.0  0.0  1.0  0.0  0.015127   0.0  0.0  1.0
```

```
train_X_unscaled, train_y = get_final_data(data_train)
```

```
scaler_train =
▾ StandardScaler
StandardScaler()
```

```
scaler_train = StandardScaler().fit(train_X_unscaled)
```

```
train_X =
889×11 Matrix{Float64}:
 -0.52731    0.0578533   0.900328  -0.56306   …   0.616794  -0.482711  -0.307941
  0.541436   0.0578533  -1.11071    1.77601      -1.62129    2.07163   -0.307941
 -0.260124  -0.561804    0.900328  -0.56306       0.616794  -0.482711  -0.307941
  0.341046   0.0578533  -1.11071    1.77601       0.616794  -0.482711  -0.307941
  0.341046  -0.561804    0.900328  -0.56306       0.616794  -0.482711  -0.307941
 -0.185971  -0.561804    0.900328  -0.56306   …  -1.62129   -0.482711   3.24738
  1.61018   -0.561804   -1.11071    1.77601       0.616794  -0.482711  -0.307941
  ⋮                                            ⋱                        ⋮
  0.608233   2.53648     0.900328  -0.56306      -1.62129   -0.482711   3.24738
 -0.193327  -0.561804   -1.11071   -0.56306       0.616794  -0.482711  -0.307941
 -0.7277    -0.561804   -1.11071    1.77601   …   0.616794  -0.482711  -0.307941
 -1.05481    1.29717     0.900328  -0.56306       0.616794  -0.482711  -0.307941
 -0.260124  -0.561804   -1.11071    1.77601      -1.62129    2.07163   -0.307941
  0.140656  -0.561804    0.900328  -0.56306      -1.62129   -0.482711   3.24738
```

```
train_X = scaler_train.transform(train_X_unscaled)
```

```
test_X_unscaled = 418×11 Matrix{Float64}:
                   34.5     0.0  1.0  0.0  0.0  1.0  0.0  0.0152816  1.0  0.0  0.0
                   47.0     0.1  1.0  0.0  0.0  0.0  1.0  0.0136631  0.0  1.0  0.0
                   62.0     0.0  0.0  1.0  0.0  1.0  0.0  0.0189087  1.0  0.0  0.0
                   27.0     0.0  1.0  0.0  0.0  1.0  0.0  0.0169081  0.0  1.0  0.0
                   22.0     0.2  1.0  0.0  0.0  0.0  1.0  0.0239836  0.0  1.0  0.0
                   14.0     0.0  1.0  0.0  0.0  1.0  0.0  0.018006   0.0  1.0  0.0
                   30.0     0.0  1.0  0.0  0.0  0.0  1.0  0.0148912  1.0  0.0  0.0
                    ⋮                            ⋮                             ⋮
                   28.0     0.0  1.0  0.0  0.0  0.0  1.0  0.0151758  0.0  1.0  0.0
                   25.7853  0.0  1.0  0.0  0.0  1.0  0.0  0.0157126  0.0  1.0  0.0
                   39.0     0.0  0.0  0.0  1.0  0.0  1.0  0.212559   0.0  0.0  1.0
                   38.5     0.0  1.0  0.0  0.0  1.0  0.0  0.0141511  0.0  1.0  0.0
                   55.3586  0.0  1.0  0.0  0.0  1.0  0.0  0.0157126  0.0  1.0  0.0
                   18.455   0.2  1.0  0.0  0.0  1.0  0.0  0.0436405  0.0  0.0  1.0
```

```
test_X_unscaled = get_test_data(data_test)
```

```
scaler_test =
▾ StandardScaler
StandardScaler()
```

```
scaler_test = StandardScaler().fit(test_X_unscaled)
```

```
test_X = 418×11 Matrix{Float64}:
  0.273794   -0.553443   0.957826  -0.534933  …   2.84376   -1.35068  -0.568142
  1.13064     0.105643   0.957826  -0.534933     -0.351647   0.74037  -0.568142
  2.15885    -0.553443  -1.04403    1.86939       2.84376   -1.35068  -0.568142
 -0.240313   -0.553443   0.957826  -0.534933     -0.351647   0.74037  -0.568142
 -0.583051    0.764728   0.957826  -0.534933     -0.351647   0.74037  -0.568142
 -1.13143    -0.553443   0.957826  -0.534933  …  -0.351647   0.74037  -0.568142
 -0.0346702  -0.553443   0.957826  -0.534933      2.84376   -1.35068  -0.568142
  ⋮                                           ⋱                        ⋮
 -0.171765   -0.553443   0.957826  -0.534933     -0.351647   0.74037  -0.568142
 -0.323581   -0.553443   0.957826  -0.534933     -0.351647   0.74037  -0.568142
  0.582259   -0.553443  -1.04403   -0.534933     -0.351647  -1.35068   1.76012
  0.547985   -0.553443   0.957826  -0.534933  …  -0.351647   0.74037  -0.568142
  1.7036     -0.553443   0.957826  -0.534933     -0.351647   0.74037  -0.568142
 -0.826054    0.764728   0.957826  -0.534933     -0.351647  -1.35068   1.76012
```

```
test_X = scaler_test.transform(test_X_unscaled)
```

Model Building

- Linear Regression
- Ridge Regression
- LASSO Regression
- Logistic Regression
- K Nearest Neighbor
- Decision Tree
- Random Forest
- Naive Bayes

```
using ScikitLearn   .CrossValidation: cross_val_score
```

Crossvalidation Scoring

Linear Regression

```
PyObject <class 'sklearn.linear_model._base.LinearRegression'>
```

```
@sk_import linear_model: LinearRegression
```

```
cv_linear =  [0.306841, 0.366066, 0.384601, 0.333992, 0.437415]
```

```
cv_linear = cross_val_score(LinearRegression(),train_X,train_y,cv=5)
```

```
print(mean(cv_linear))
```

0.36578285815718764

Ridge Regression

PyObject <class 'sklearn.linear_model._ridge.Ridge'>

```
@sk_import linear_model: Ridge
```

cv_ridge = [0.306868, 0.366154, 0.384587, 0.334045, 0.437383]

```
cv_ridge = cross_val_score(Ridge(),train_X,train_y,cv=5)
```

```
print(mean(cv_ridge))
```

0.36580750089074376

LASSO Regression

PyObject <class 'sklearn.linear_model._coordinate_descent.Lasso'>

```
@sk_import linear_model: Lasso
```

cv_lasso = [-0.0228073, -0.0238521, -0.000177285, -0.00315337, -0.00708916]

```
cv_lasso = cross_val_score(Lasso(),train_X,train_y,cv=5)
```

```
print(mean(cv_lasso))
```

-0.011415839277551543

Logistic Regression

PyObject <class 'sklearn.linear_model._logistic.LogisticRegression'>

```
@sk_import linear_model: LogisticRegression
```

cv_logistic = [0.775281, 0.786517, 0.780899, 0.775281, 0.819209]

```
cv_logistic = cross_val_score(LogisticRegression(),train_X,train_y,cv=5)
```

```
print(mean(cv_logistic))
```

0.7874373135275821

K Nearest Neighbor

PyObject <class 'sklearn.neighbors._classification.KNeighborsClassifier'>

```
@sk_import neighbors: KNeighborsClassifier
```

cv_kneighbors = [0.769663, 0.786517, 0.808989, 0.842697, 0.79096]

```
cv_kneighbors = cross_val_score(KNeighborsClassifier(),train_X,train_y,cv=5)
```

```
print(mean(cv_kneighbors))
```

0.7997651241033454

Decision Tree

PyObject <class 'sklearn.tree._classes.DecisionTreeClassifier'>

```
@sk_import tree: DecisionTreeClassifier
```

cv_destree = [0.696629, 0.780899, 0.803371, 0.769663, 0.745763]

```
cv_destree = cross_val_score(DecisionTreeClassifier(),train_X,train_y,cv=5)
```

```
print(mean(cv_destree))
```

0.7592649019234431

Random Forest

PyObject <class 'sklearn.ensemble._forest.RandomForestClassifier'>

```
@sk_import ensemble: RandomForestClassifier
```

cv_randforest = [0.752809, 0.780899, 0.859551, 0.797753, 0.80226]

```
cv_randforest = cross_val_score(RandomForestClassifier(),train_X,train_y,cv=5)
```

```
print(mean(cv_randforest))
```

0.7986542245921411

Naive Bayes

PyObject <class 'sklearn.naive_bayes.GaussianNB'>

```
@sk_import naive_bayes: GaussianNB
```

cv_naibay = [0.752809, 0.775281, 0.780899, 0.820225, 0.813559]

```
cv_naibay = cross_val_score(GaussianNB(),train_X,train_y,cv=5)
```

```
print(mean(cv_naibay))
```

0.7885545610359932

Voting Classifier

PyObject <class 'sklearn.ensemble._voting.VotingClassifier'>

```
@sk_import ensemble: VotingClassifier
```

voting_clf_soft =

VotingClassifier

lr	knn	dt	
LogisticRegression	KNeighborsClassifier	DecisionTreeClassifier	Rando

```
voting_clf_soft = VotingClassifier(estimators = [
    ("lr", LogisticRegression()),
    ("knn", KNeighborsClassifier()),
    ("dt", DecisionTreeClassifier()),
    ("rf", RandomForestClassifier()),
    ("gnb", GaussianNB())],
    voting = "soft")
```

```
voting_clf_hard = VotingClassifier(estimators = [
    ("lr", LogisticRegression()),
    ("knn", KNeighborsClassifier()),
    ("dt", DecisionTreeClassifier()),
    ("rf", RandomForestClassifier()),
    ("gnb", GaussianNB())],
    voting = "hard");
```

cv_soft = [0.764045, 0.797753, 0.853933, 0.814607, 0.830508]

```
cv_soft = cross_val_score(voting_clf_soft,train_X,train_y,cv=5)
```

cv_hard = [0.786517, 0.808989, 0.848315, 0.831461, 0.830508]

```
cv_hard = cross_val_score(voting_clf_hard,train_X,train_y,cv=5)
```

```
print(mean(cv_soft))
```

0.8121691106455913

```
print(mean(cv_hard))
```

0.8211578746905349

Model Fitting

model =

▸ VotingClassifier

lr	knn	dt	
▸ LogisticRegression	▸ KNeighborsClassifier	▸ DecisionTreeClassifier	▸ Randoı

```
model = fit!(voting_clf_soft,train_X,train_y)
```

y_pred =

[0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, more ,1, 1, 0, 1, 1, 0, 1, 0,

```
y_pred = predict(model,test_X)
```

df_submission =

	PassengerId	Survived
1	892	0
2	893	0
3	894	0
4	895	0
5	896	1
6	897	0
7	898	0
8	899	0
9	900	1
10	901	0
more		
418	1309	0

```
df_submission = DataFrame("PassengerId" => 892:1309,
                          "Survived" => y_pred)
```

submissionfile = "submission.csv"

```
submissionfile = "submission.csv"
```

"submission.csv"

```
CSV.write(submissionfile, df_submission)
```