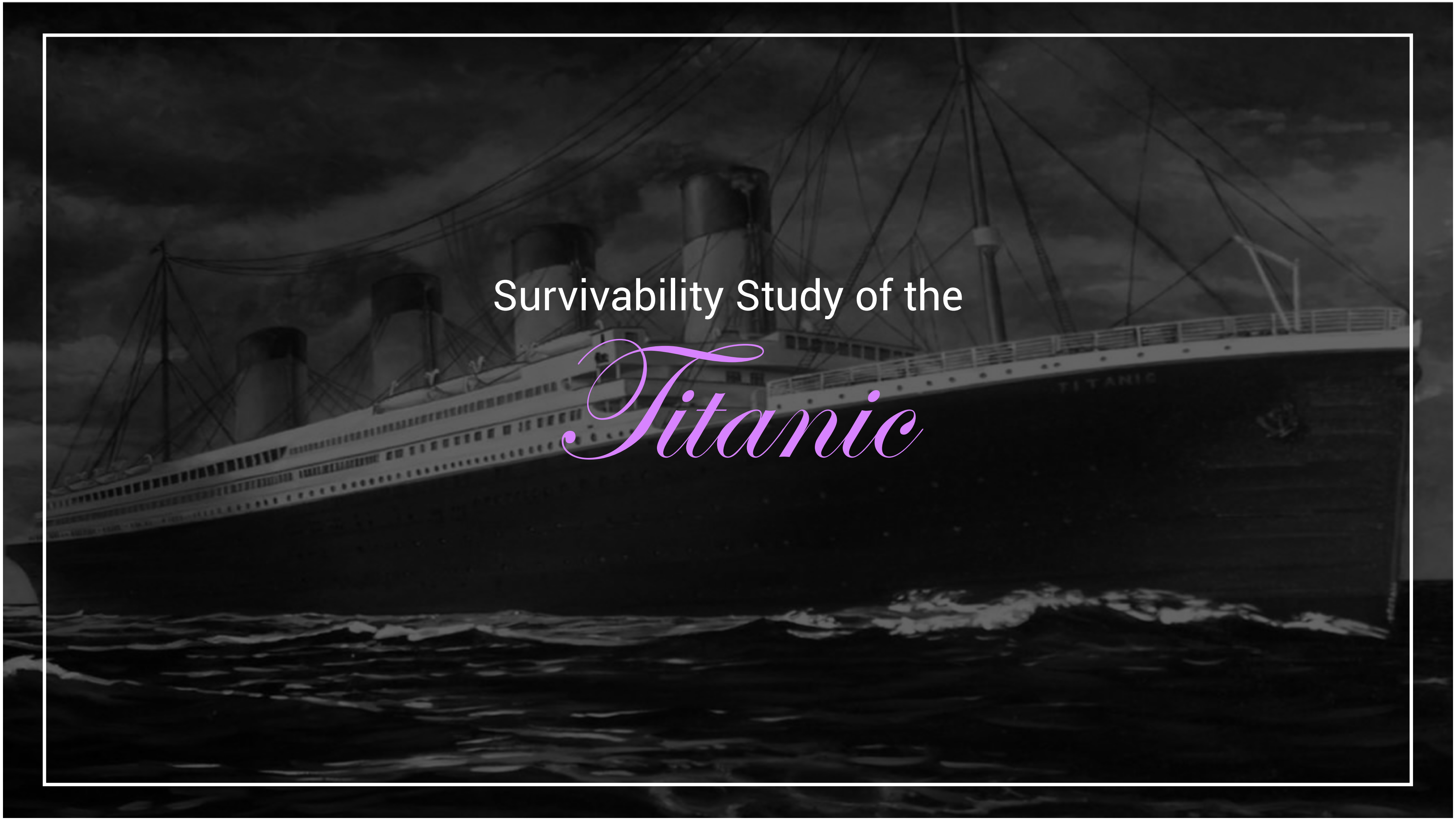


# Survivability Study of the *Titanic*

# RMS Titanic sank during her maiden voyage on 15 Apr 1912. Only 42% survived.

### MAIDEN VOYAGE WAS NEVER COMPLETED

The largest superliner of the time sunk after hitting an iceburg during her maiden voyage from Southampton to New York.

### ONLY 20 LIFEBOATS

The lifeboats were only sufficient for one third of her total capacity due to the maritime safety regulations in the days.

### LIFEBOATS WERE HALF-FILLED

At the time of the sinking, the lowered lifeboats were only about half-filled.

### CHANCES OF SURVIVAL WERE NOT EQUAL

A disproportionate number of men were left aboard because of a "women and children first" protocol for loading lifeboats.

## THE CHALLENGE

Build a predictive model that answers the question: “what sorts of people were more likely to survive?” using passenger data (i.e. name, age, gender, socio-economic class, etc.).


1. Exploratory Data Analysis

# Overview of the Titanic dataset

Training and testing datasets were split by PassengerId – not completely random.

Columns with missing values:

1. Cabin – 77% missing
2. Age – 20% missing
3. Embarked – 2 rows missing

## 2.1 Describe dataset

There are 891 observations (rows) and 12 variables (columns).

**Identifiers**

1. `PassengerId` Unique ID of passenger
2. `Name` Passenger name

**Categorical variables**

1. `Survived` Survival. 0 = No, 1 = Yes
2. `Pclass` Ticket class. 1 = 1st (Upper SES), 2 = 2nd (Middle SES), 3 = 3rd (Lower SES)
3. `Sex` Sex
4. `Ticket` Ticket number
5. `Cabin` Cabin number. The first letter denotes the deck.
6. `Embarked` Port of embarkation. C = Cherbourg, Q = Queenstown, S = Southampton

**Numerical variables**

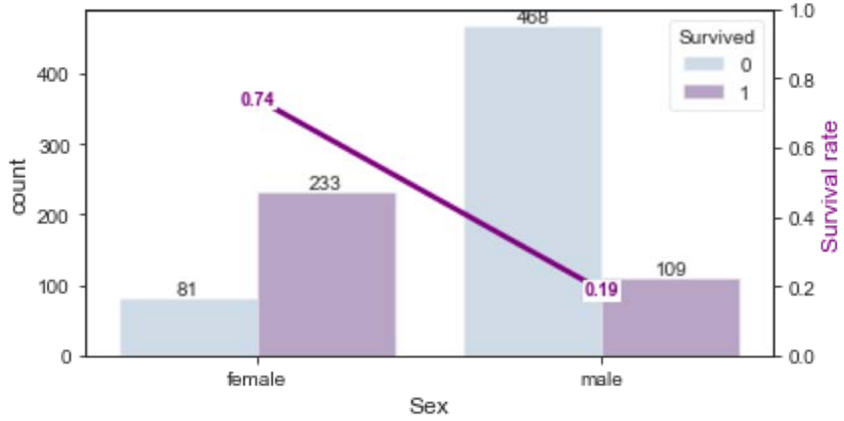
1. `Age` Age in years
2. `Sibsp` # of siblings / spouses on board
3. `Parch` # of parents / children on board
4. `Fare` Passenger fare

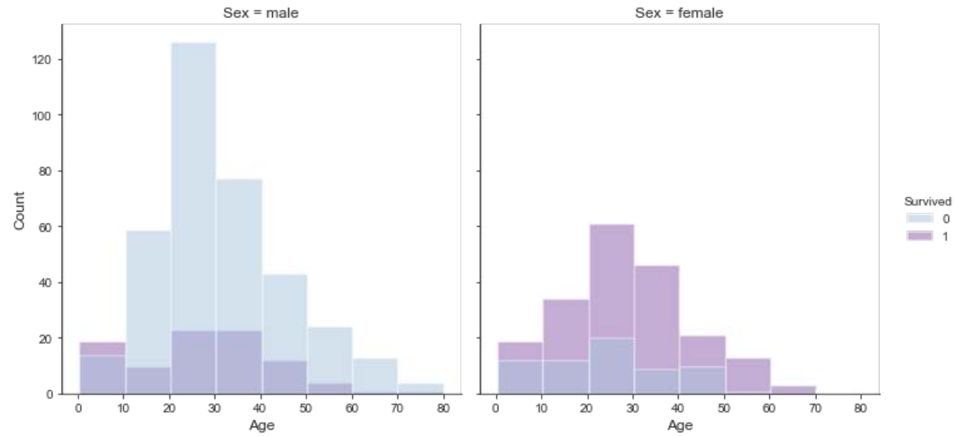
1. Exploratory Data Analysis

# Survival rate for males was only 20%

That's about one-third of the survival rate of females.

Survival rate for males was particularly low for ages aged 20 to 30.

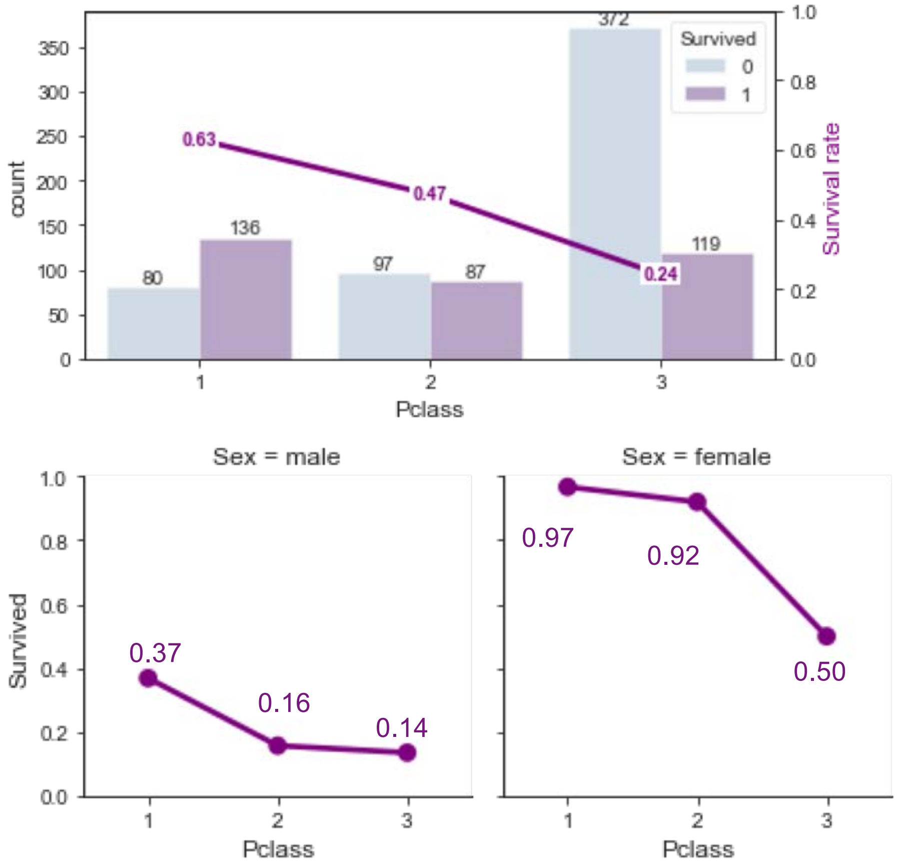


1. Exploratory Data Analysis

# Survival rate was higher for higher Passenger Classes

The difference in survival rate is amplified when we consider Sex.

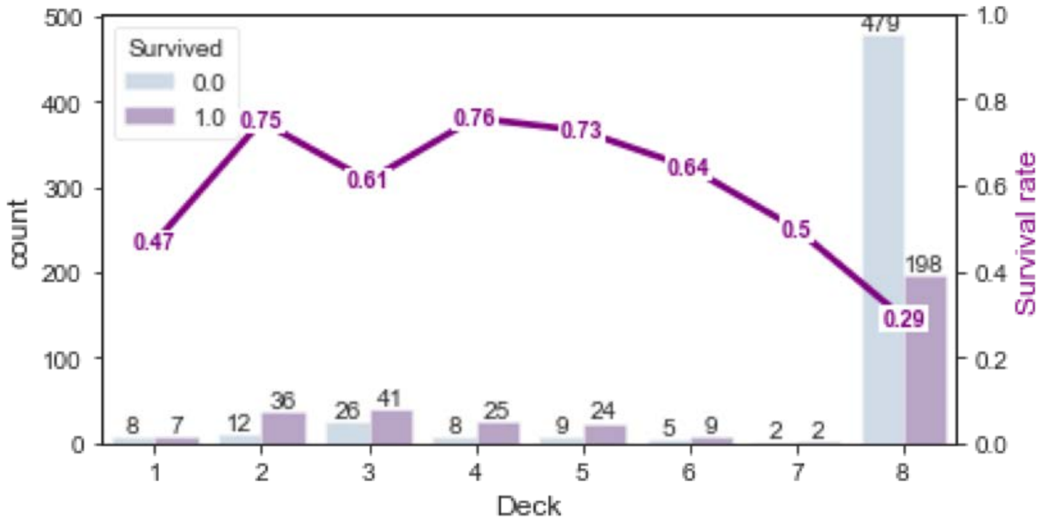


1. Exploratory Data Analysis

# Deck appears to be related to survival, but may not be generalisable.

Extracting the first letter of the cabin gives the deck. We map the letter to the number of levels below the main deck.

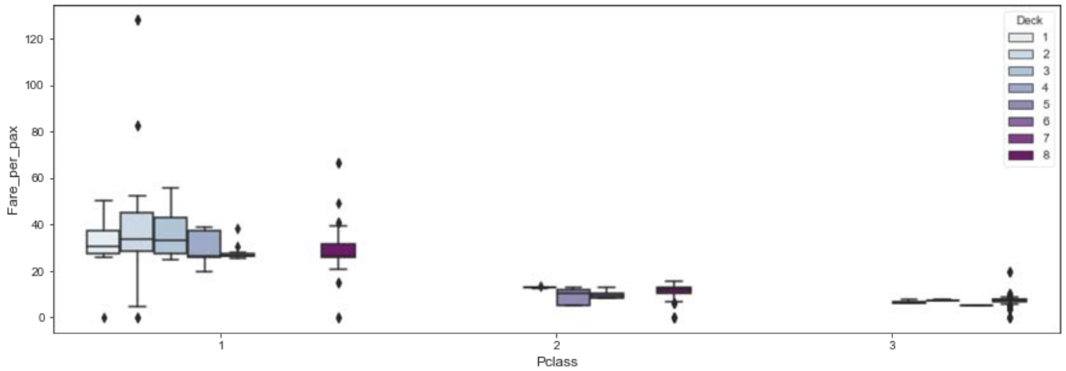
It appears that the probability of survival is higher for at Deck = 2, 4, and 5, but there is a good portion of data missing (i.e. Deck = 8).



# Can Fare be used to derive information about deck?

Deck may be a contributor to survival – passengers on higher decks could have reached the lifeboats first. We explore if Deck can be estimated from Fare.

Since Fare is a sum, we create a Fare per pax variable.




Similarity in the distributions (overlapping boxplots) show that Fare per pax is not a good indicator of Deck, but is a good indicator of Pclass.

# 2. Data Preparation

# We try to preserve as much info in the data cleaning process

## CABIN

There are too many missing values to perform a meaningful substitution. Hence, we will use Pclass and Deck as high-level proxies for Cabin.

## EMBARKED

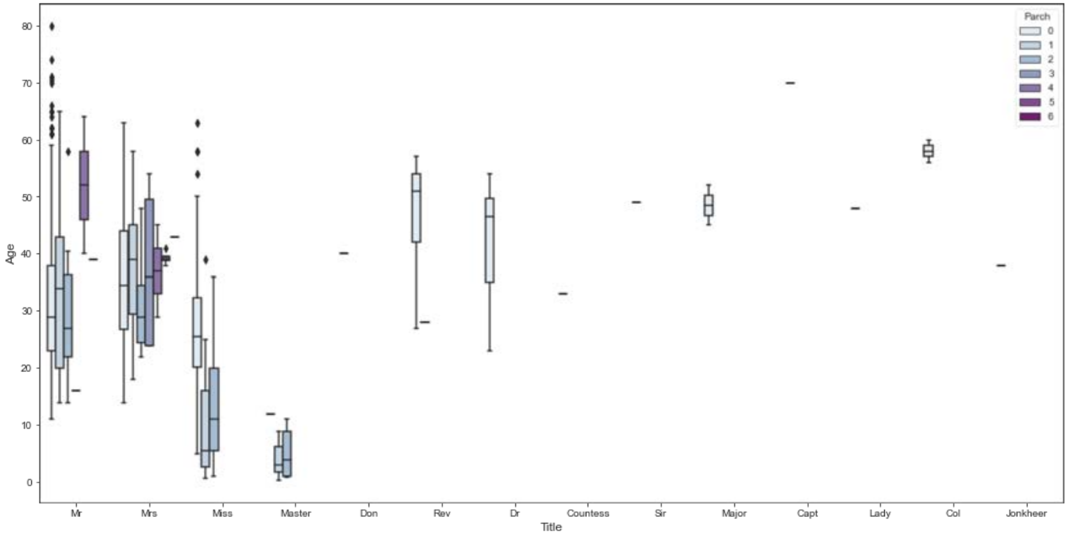
There are 2 missing values and both hold the same ticket. We will use mode substitution.

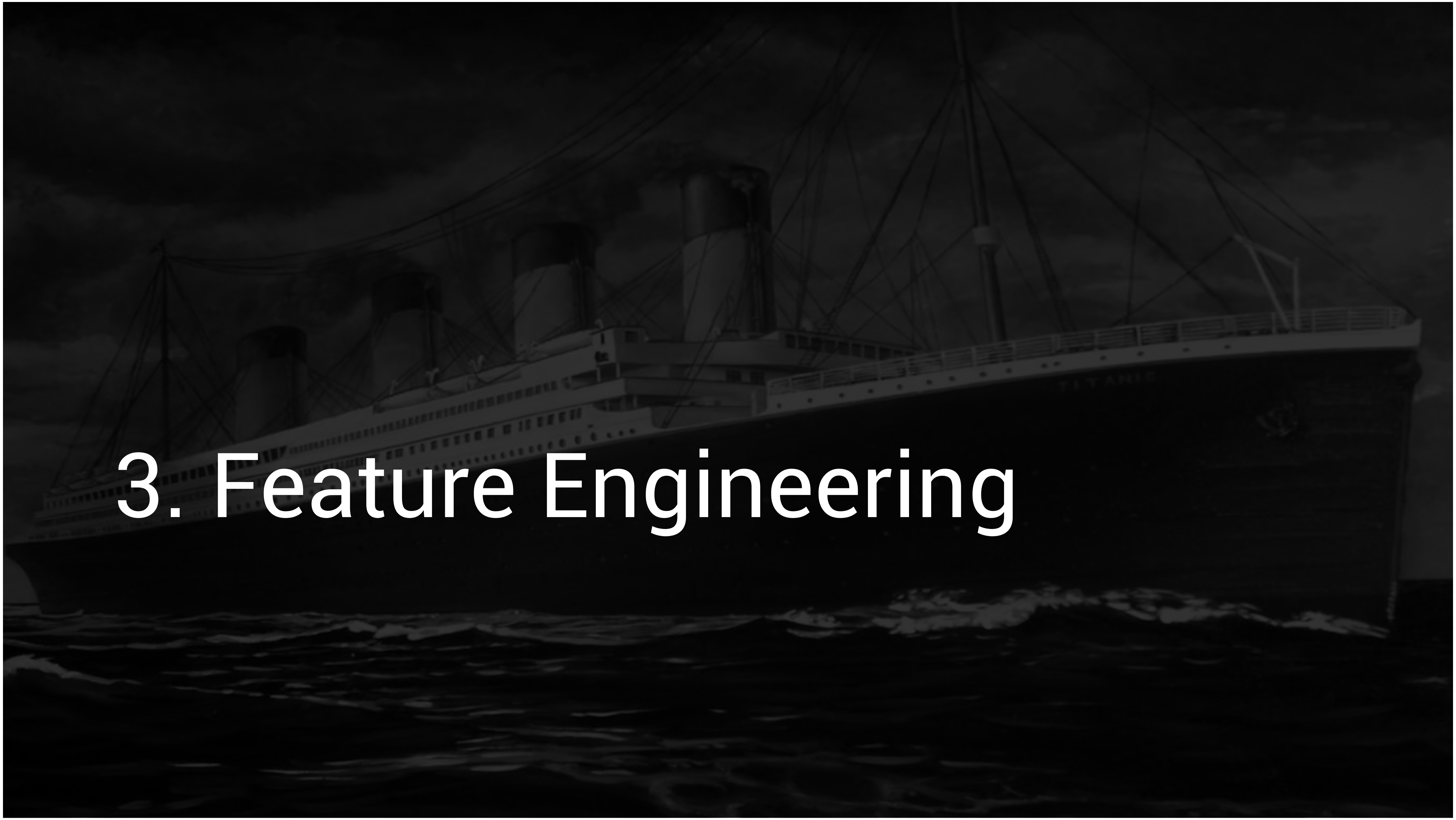
MORE >>

# AGE: Impute using median substitution, partitioned by Title and Parch

We considered

1. Age is differently distributed across different Title-Parch groups, some more significantly different (when boxplots do not overlap).
2. Age is skewed.




3. Feature Engineering

# More features...

### EMBARKED_ORDINAL

After doing one hot-encoding for Embarked, we create a column based on the order of embarkation (i.e. S=1, C = 2, Q=3).

### TITLE and TITLE_GROUP

We extracted Title from name, and also grouped them into 6 categories ( 'Mr' , 'Mrs', 'Miss', 'Master' , 'Reputable_Male', 'Reputable_Female') to reduce noise.

### REPUTABLE

We classify those that have honorific Titles as Reputable, since these people may have been accorded more privileges which could increase their chances of survival.

### RELATIVES and ALONE

To reduce noise, we sum SibSp and Parch to obtain the number of Relatives onboard. Alone indicates lone travellers.

3. Feature Engineering

# Even more features…

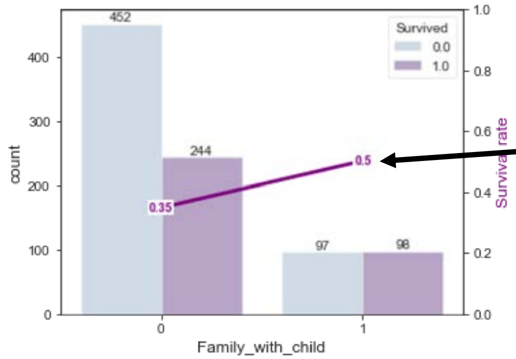
1. **FAMILY_WITH_CHILD**

   It's possible that families with children have better survival rates, since there may have been a guardian/parent accompanying the child on the lifeboat.

2. **FAMILY_WITH_SURVIVAL**

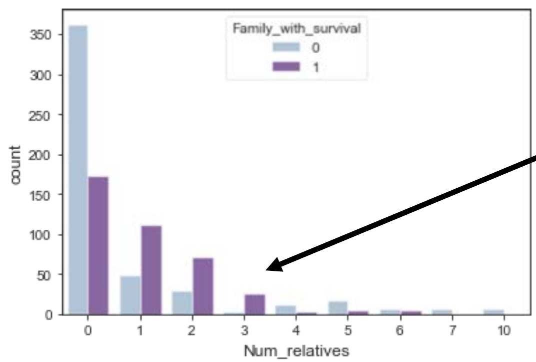
   For similar reasons as above, perhaps people don't want to be separated and maybe survivors will be accompanied.

**Survival rate for families with children**




Survival rate is better for families with children (below 16 years old)

**Survival patterns for families with survivors**



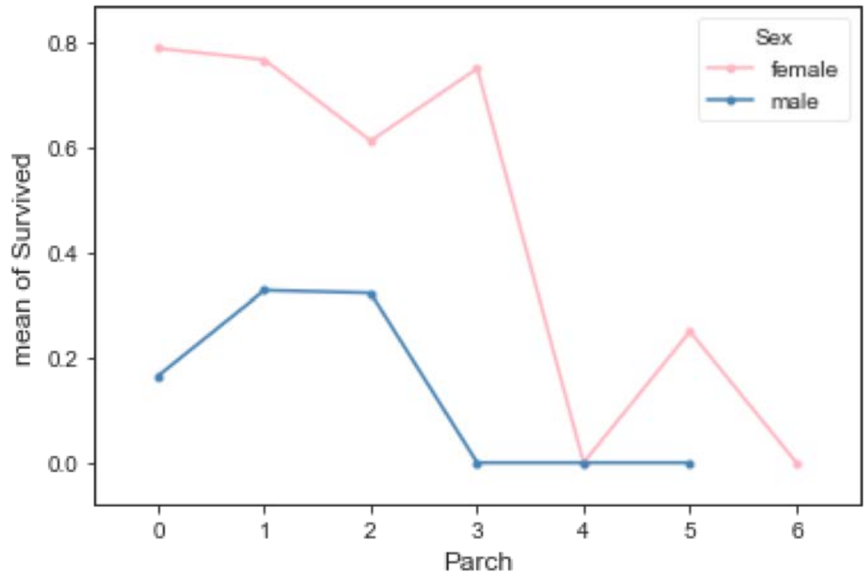

Families around 2 to 4 persons are more likely to have at least 1 survivor

3. Feature Engineering

# Exploring interactions between Sex and Parch

Sex is a strong predictor. Sex may have some interaction effects on Parch (non-parallel lines). Hence we create the following interaction variables

1. Male_Parch (male = 1 * Parch)
2. Female_Parch (female = 1 * Parch)
3. Sex_Parch (male = 1 * Parch, female = -1 * Parch)



# 4. Modelling

# For each model, we…

### GENERATED FEATURE IMPORTANCES

Different models produce a different ranked list of variables, though mostly similar.

### PERFORM BACKWARD SELECTION OF FEATURES

We generated the full model, then eliminate features that are least important or have high multicollinearity.

### TRAIN-TEST SPLIT = 80:20

Set seed = 21

### PERFORMED STANDARD SCALING WHEN NEEDED

Used standard scaling for SVC, Logistic Regression, Naïve Bayes, XGBoost

### DID CROSS VALIDATION AND HYPERPARAMETER TUNING IN EVERY RUN

Using GridSearchCV, cv = 5. Because FOMO.

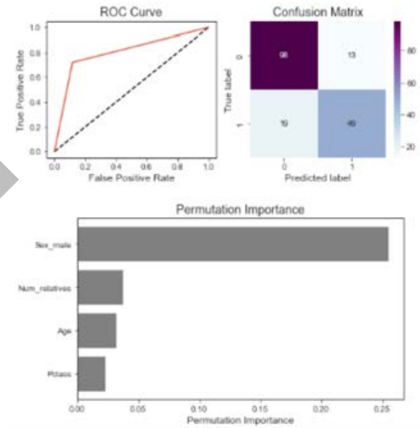
# Here's what went on behind the scenes…

## From code

```
# Define function to generate metrics for classification

def classification_metrics(cv):
    print("*"*60)
    modelname = [item for item in steps if item[0]=='model'][0][1]
    print("METHOD: {}".format(modelname))
    print("*"*60,end='\n\n')

    print("Features: {}".format(X.columns.tolist()))
    print("Standardized features: {}".format(to_scale))
    print("Tuned Model Parameters: {}".format(cv.best_params_))

    # Metrics
    print('\n')
    print("========== Metrics ==========" ,end='\n\n')
    print("Accuracy: {:.3f}".format(cv.score(X_test, y_test)))
    print("AUC: {:.3f}".format(metrics.roc_auc_score(y_test, y_pred)))
    r2 = cv.score(X_test, y_test)
    print("R²: {:.3f}".format(r2))
    r2_adj = 1 - ( 1-cv.score(X, y) ) * ( len(y) - 1 ) / ( len(y) - X.shape[1] - 1 )
    print("Adjusted R²: {:.3f}".format(r2_adj))

    print('\n')
    print("=== Classification Report ===" ,end='\n\n')
    print(metrics.classification_report(y_test, y_pred))

    # Plots
    fig = plt.figure(figsize=(8,3))

    # ROC curve
    plt.subplot(1,2,1)
    fpr, tpr, thresholds = metrics.roc_curve(y_test, y_pred)
    plt.plot([0, 1], [0, 1], 'k--')
    plt.plot(fpr, tpr)
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')

    # Confusion matrix
    plt.subplot(1,2,2)
    cm = metrics.confusion_matrix(y_test, y_pred, labels=cv.classes_)
    sns.heatmap(cm, annot=True, cmap="BuPu")
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted label')
    plt.ylabel('True label')
    plt.show()

# ----------------------------------------------------------------------
# Show feature importances by coef
def plot_feature_importance_coef(cv):
    features = X.columns.tolist()
    importances = cv.best_estimator_.named_steps['model'].coef_

    feturedf=pd.DataFrame({'Features':features,'Importances':importances})
    feturedf.sort_values('Importances',inplace=True)

    fig = plt.figure(figsize=(7,4))
    feturedf.plot(kind='barh',                ,x='Features',color='gray')
    plt.xlabel("Coeffic
    plt.title(
    plt.
```

## To report

```
************************************************************
METHOD: SVC()
************************************************************

Features: ['Pclass', 'Age', 'Num_relatives', 'Sex_male']
Standardized features: ['Pclass', 'Age', 'Num_relatives', 'Sex_male']
Tuned Model Parameters: {'model__C': 1, 'model__gamma': 0.1, 'model__kernel': 'rbf'}


========== Metrics ==========

Accuracy: 0.821
AUC: 0.802
R²: 0.821
Adjusted R²: 0.828


=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.84      0.88      0.86       111
         1.0       0.79      0.72      0.75        68

    accuracy                           0.82       179
   macro avg       0.81      0.80      0.81       179
weighted avg       0.82      0.82      0.82       179
```



# Best ways to improve model scores

1

## SELECTING FEATURES

Choosing correct features has the most significant improvement to model scores

2

## SELECTING MODEL

Second most significant method to improve scores

4. Modelling

# Family_with_ survival

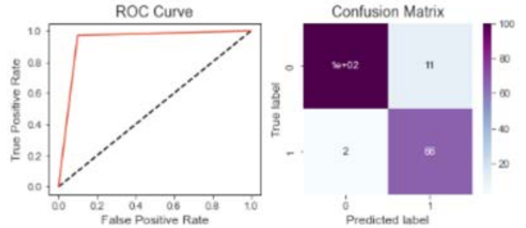
Strong predictor for ALL training models (accuracy > 0.9!), but testing scores drop (>0.75).

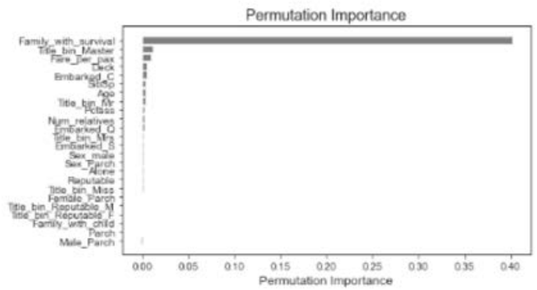
Why? That's because the Kaggle dataset is split by order of PassengerId, and the test set contains families the training set has not seen before.

This variable also "suppresses" the others. So with this behavior, we shall drop if from the models.

Classification results from SVC model that are too good to be true

```
========== Metrics ==========

Accuracy: 0.927
AUC: 0.936
R²: 0.927
Adjusted R²: 0.958

=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.98      0.90      0.94       111
         1.0       0.86      0.97      0.91        68

    accuracy                           0.93       179
   macro avg       0.92      0.94      0.92       179
weighted avg       0.93      0.93      0.93       179
```



**Is there value then to create a variable indicating if the whole family survived?**

Maybe, but then we are creating a variable from a known outcome for prediction. Which is circular and is not truly predicting.

So we don't do this.

# A systematic way to choose features

### 1. KEEP ONLY BEST OF CORRELATED FEATURES

Since we have many derived features, we only keep the best. Dropping features, especially those high in feature importance, would have an impact on the feature ranks.

*e.g. Title_group was derived from Sex and Reputable. Since Reputable has little or no impact to the model, we drop Reputable and its derivative Title_group.*

### 2. TRY REPLACING CATEGORICAL FEATURES WITH NUMERICAL, AND VICE VERSA

Sometimes the features have too much/little noise, and we trial-and-error to see which type of features work best for the model.

*e.g. Choosing either Pclass or Fare_per_pax*

### 3. DROP REMAINING FEATURES THAT HAVE VERY LOW IMPORTANCE SCORES/COEFFICIENTS

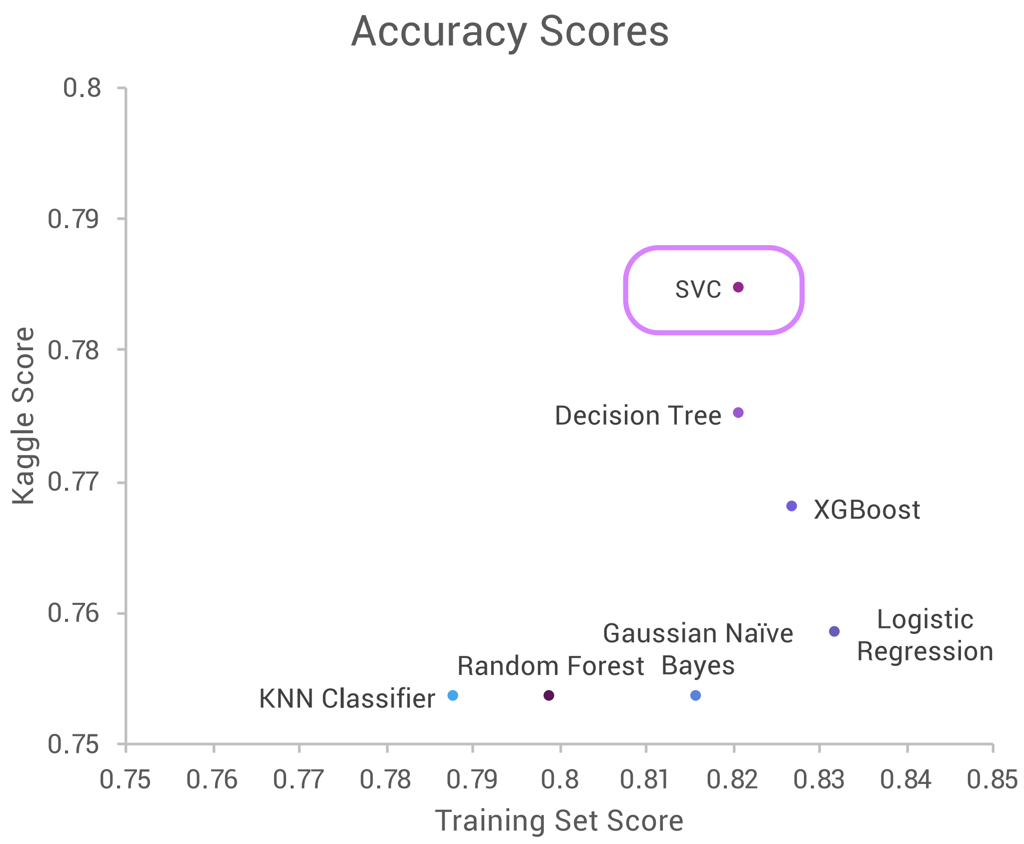
As the final step, we can safely drop insignificant features.

4. Modelling

# SVC was the best

**78% accuracy on testing data**

After lots of ~~trial-end-error~~ systematic experimenting



4. Modelling

# SVC was the simplest

With only 4 features

| Models | Pclass | Reputable | Age | Sex | Deck | Fare_per_pax | Num_relatives | SibSp | Parch | Sex_Parch |
|---|---|---|---|---|---|---|---|---|---|---|
| SVC | ✓ | | ✓ | ✓ | | | ✓ | | | |
| Decision Tree | ✓ | | ✓ | ✓ | ✓ | | ✓ | | | |
| XGBoost | | | ✓ | ✓ | ✓ | | ✓ | ✓ | | |
| Logistic Regression | ✓ | | ✓ | ✓ | ✓ | | | | ✓ | ✓ |
| KNN Classifier | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | | | |
| Gaussian Naïve Bayes | ✓ | | ✓ | ✓ | | | ✓ | | | |
| Random Forest | | | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | |

## IN SUMMARY

# “What sorts of people were more likely to survive?”

Based on the SVC model selected (78% accuracy), the following are more likely to survive

1. Females
2. Younger persons
3. Those with more relatives on board
4. Those in a higher Pclass

4. Summary

# Some takeaways from this exercise…

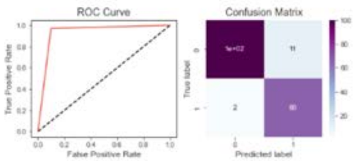
1. Feature selection has the most significant impact on the model.
2. Strong predictors aren't always a good thing.
3. Concise and easy-to-implement models are the best.
4. The models generated may have some overfitting since the testing scores dropped by ~4 percentage points, but this is acceptable.
5. There's room to explore stacking methods too.

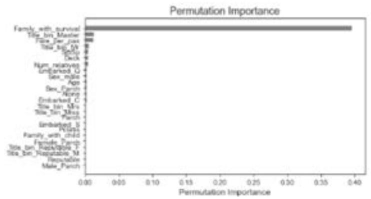

Annex
All the hard work.

# Support Vector Classification

## Full Model

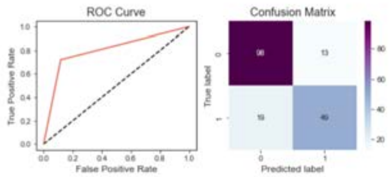
```
************************************************************
METHOD: SVC()
************************************************************

Features: ['Pclass', 'Age', 'SibSp', 'Parch', 'Deck', 'Fare_per_pax', 'Sex_male', 'Embarked_C', 'Embarked_Q', '
Embarked_S', 'Title_bin_Master', 'Title_bin_Miss', 'Title_bin_Mr', 'Title_bin_Mrs', 'Title_bin_Reputable_F', 'T
itle_bin_Reputable_M', 'Reputable', 'Num_relatives', 'Alone', 'Family_with_child', 'Family_with_survival', 'Mal
e_Parch', 'Female_Parch', 'Sex_Parch']
Standardized features: ['Pclass', 'Age', 'SibSp', 'Parch', 'Deck', 'Fare_per_pax', 'Sex_male', 'Embarked_C', 'E
mbarked_Q', 'Embarked_S', 'Title_bin_Master', 'Title_bin_Miss', 'Title_bin_Mr', 'Title_bin_Mrs', 'Title_bin_Rep
utable_F', 'Title_bin_Reputable_M', 'Reputable', 'Num_relatives', 'Alone', 'Family_with_child', 'Family_with_su
rvival', 'Male_Parch', 'Female_Parch', 'Sex_Parch']
Tuned Model Parameters: {'model__C': 1, 'model__gamma': 0.1, 'model__kernel': 'linear'}

========== Metrics ==========

Accuracy: 0.927
AUC: 0.936
R²: 0.927
Adjusted R²: 0.958

=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.98      0.90      0.94       111
         1.0       0.86      0.97      0.91        68

    accuracy                           0.93       179
   macro avg       0.92      0.94      0.92       179
weighted avg       0.93      0.93      0.93       179
```

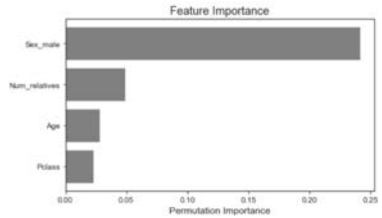



## Best Model

```
************************************************************
METHOD: SVC()
************************************************************

Features: ['Pclass', 'Age', 'Num_relatives', 'Sex_male']
Standardized features: ['Age']
Tuned Model Parameters: {'SVM__C': 1, 'SVM__gamma': 0.1, 'SVM__kernel': 'rbf'}

========== Metrics ==========

Accuracy: 0.821
AUC: 0.802
R²: 0.821
Adjusted R²: 0.826

=== Classification Report ===

              precision    recall  f1-score   support

           0       0.84      0.88      0.86       111
           1       0.79      0.72      0.75        68

    accuracy                           0.82       179
   macro avg       0.81      0.80      0.81       179
weighted avg       0.82      0.82      0.82       179
```

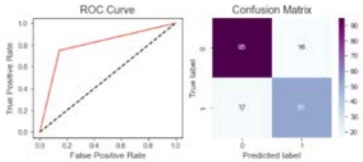


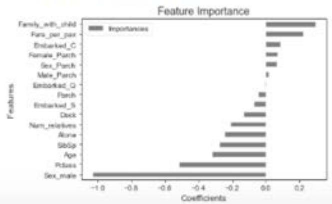
# Logistic Regression

## Full Model

```
************************************************************
METHOD: LogisticRegression()
************************************************************

Features: ['Pclass', 'Age', 'SibSp', 'Parch', 'Deck', 'Fare_per_pax', 'Sex_male', 'Embarked_C', 'Embarked_Q', '
Embarked_S', 'Num_relatives', 'Alone', 'Family_with_child', 'Male_Parch', 'Female_Parch', 'Sex_Parch']
Standardized features: ['Pclass', 'Age', 'SibSp', 'Parch', 'Deck', 'Fare_per_pax', 'Sex_male', 'Embarked_C', 'E
mbarked_Q', 'Embarked_S', 'Num_relatives', 'Alone', 'Family_with_child', 'Male_Parch', 'Female_Parch', 'Sex_Par
ch']
Tuned Model Parameters: {'model__C': 0.046415888336127774, 'model__penalty': 'l2'}


========== Metrics ==========

Accuracy: 0.816
AUC: 0.803
R²: 0.816
Adjusted R²: 0.809


=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.85      0.86      0.85       111
         1.0       0.76      0.75      0.76        68

    accuracy                           0.82       179
   macro avg       0.80      0.80      0.80       179
weighted avg       0.82      0.82      0.82       179
```

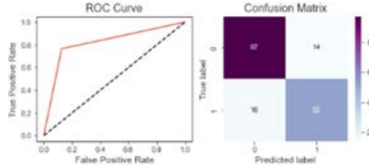



```
<Figure size 504x288 with 0 Axes>
```

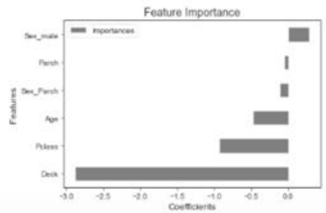



## Best Model

```
************************************************************
METHOD: LogisticRegression()
************************************************************

Features: ['Pclass', 'Age', 'Parch', 'Sex_male', 'Sex_Parch', 'Deck']
Standardized features: ['Pclass', 'Age', 'Parch', 'Sex_Parch', 'Deck']
Tuned Model Parameters: {'model__C': 21.54434690031882, 'model__penalty': 'l2'}


========== Metrics ==========

Accuracy: 0.832
AUC: 0.819
R²: 0.832
Adjusted R²: 0.791


=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.86      0.87      0.87       111
         1.0       0.79      0.76      0.78        68

    accuracy                           0.83       179
   macro avg       0.82      0.82      0.82       179
weighted avg       0.83      0.83      0.83       179
```



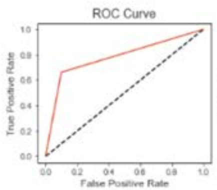

```
<Figure size 504x288 with 0 Axes>
```

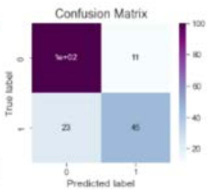


# K-Nearest Neighbours

## Full Model

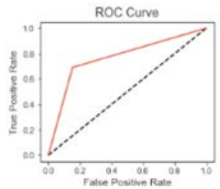
```
************************************************************
METHOD: KNeighborsClassifier()
************************************************************

Features: ['Pclass', 'Age', 'SibSp', 'Parch', 'Deck', 'Fare_per_pax', 'Sex_male', 'Embarked_C', 'Embarked_Q', '
Embarked_S', 'Title_bin_Master', 'Title_bin_Miss', 'Title_bin_Mr', 'Title_bin_Mrs', 'Title_bin_Reputable_F', 'T
itle_bin_Reputable_M', 'Reputable', 'Num_relatives', 'Alone', 'Family_with_child', 'Male_Parch', 'Female_Parch'
, 'Sex_Parch']
Standardized features: ['Pclass', 'Age', 'SibSp', 'Parch', 'Deck', 'Fare_per_pax', 'Sex_male', 'Embarked_C', 'E
mbarked_Q', 'Embarked_S', 'Title_bin_Master', 'Title_bin_Miss', 'Title_bin_Mr', 'Title_bin_Mrs', 'Title_bin_Rep
utable_F', 'Title_bin_Reputable_M', 'Reputable', 'Num_relatives', 'Alone', 'Family_with_child', 'Male_Parch', '
Female_Parch', 'Sex_Parch']
Tuned Model Parameters: {'model__n_neighbors': 15, 'model__weights': 'uniform'}


========== Metrics ==========

Accuracy: 0.810
AUC: 0.781
R²: 0.810
Adjusted R²: 0.829


=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.81      0.90      0.85       111
         1.0       0.80      0.66      0.73        68

    accuracy                           0.81       179
   macro avg       0.81      0.78      0.79       179
weighted avg       0.81      0.81      0.81       179
```

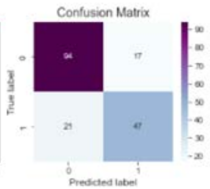



## Best Model

```
************************************************************
METHOD: KNeighborsClassifier()
************************************************************

Features: ['Pclass', 'Age', 'Num_relatives', 'Fare_per_pax', 'Sex_male', 'Reputable']
Standardized features: ['Pclass', 'Age', 'Num_relatives', 'Fare_per_pax', 'Sex_male', 'Reputable']
Tuned Model Parameters: {'model__n_neighbors': 24, 'model__weights': 'distance'}


========== Metrics ==========

Accuracy: 0.788
AUC: 0.769
R²: 0.788
Adjusted R²: 0.947


=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.82      0.85      0.83       111
         1.0       0.73      0.69      0.71        68

    accuracy                           0.79       179
   macro avg       0.78      0.77      0.77       179
weighted avg       0.79      0.79      0.79       179
```

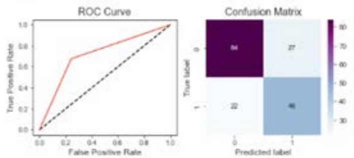


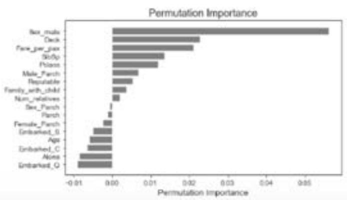



# Gaussian Naïve Bayes

## Full Model

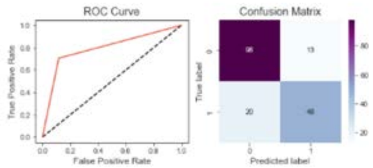
```
************************************************************
METHOD: GaussianNB()
************************************************************

Features: ['Pclass', 'Age', 'SibSp', 'Parch', 'Deck', 'Fare_per_pax', 'Sex_male', 'Embarked_C', 'Embarked_Q', '
Embarked_S', 'Reputable', 'Num_relatives', 'Alone', 'Family_with_child', 'Male_Parch', 'Female_Parch', 'Sex_Par
ch']
Standardized features: ['Pclass', 'Age', 'SibSp', 'Parch', 'Deck', 'Fare_per_pax', 'Sex_male', 'Embarked_C', 'E
mbarked_Q', 'Embarked_S', 'Reputable', 'Num_relatives', 'Alone', 'Family_with_child', 'Male_Parch', 'Female_Par
ch', 'Sex_Parch']
Tuned Model Parameters: {'model__priors': None, 'model__var_smoothing': 0.006579332246575682}

========== Metrics ==========

Accuracy: 0.726
AUC: 0.717
R²: 0.726
Adjusted R²: 0.743

=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.79      0.76      0.77       111
         1.0       0.63      0.68      0.65        68

    accuracy                           0.73       179
   macro avg       0.71      0.72      0.71       179
weighted avg       0.73      0.73      0.73       179
```

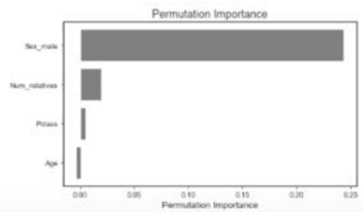



## Best Model

```
************************************************************
METHOD: GaussianNB()
************************************************************

Features: ['Pclass', 'Age', 'Num_relatives', 'Sex_male']
Standardized features: ['Pclass', 'Age', 'Num_relatives', 'Sex_male']
Tuned Model Parameters: {'model__priors': None, 'model__var_smoothing': 0.43287612810830584}

========== Metrics ==========

Accuracy: 0.816
AUC: 0.794
R²: 0.816
Adjusted R²: 0.804

=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.83      0.88      0.86       111
         1.0       0.79      0.71      0.74        68

    accuracy                           0.82       179
   macro avg       0.81      0.79      0.80       179
weighted avg       0.81      0.82      0.81       179
```

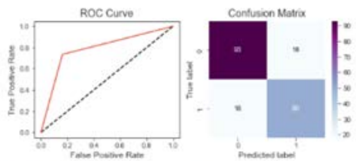


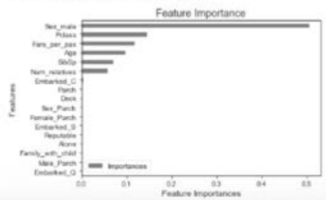
# Decision Tree

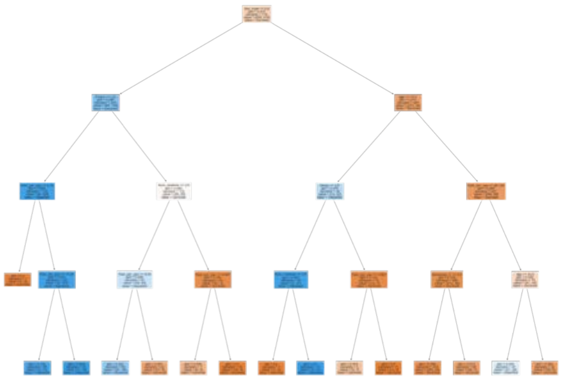
## Full Model

```
****************************************************************
METHOD: DecisionTreeClassifier()
****************************************************************

Features: ['Pclass', 'Age', 'SibSp', 'Parch', 'Deck', 'Fare_per_pax', 'Sex_male', 'Embarked_C', 'Embarked_Q', '
Embarked_S', 'Reputable', 'Num_relatives', 'Alone', 'Family_with_child', 'Male_Parch', 'Female_Parch', 'Sex_Par
ch']
Standardized features: [None]
Tuned Model Parameters: {'criterion': 'gini', 'max_depth': 4}

========== Metrics ==========

Accuracy: 0.799
AUC: 0.787
R²: 0.799
Adjusted R²: 0.848

=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.84      0.84      0.84       111
         1.0       0.74      0.74      0.74        68

    accuracy                           0.80       179
   macro avg       0.79      0.79      0.79       179
weighted avg       0.80      0.80      0.80       179
```



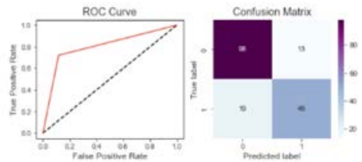

```
<Figure size 504x288 with 0 Axes>
```



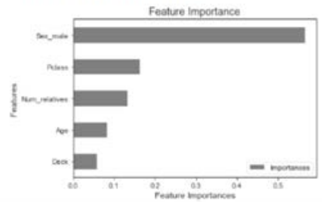



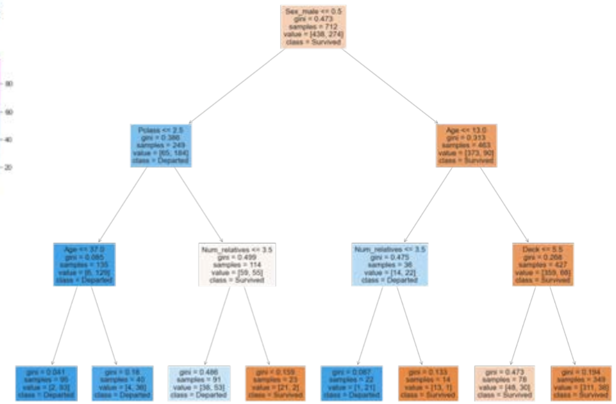
## Best Model

```
****************************************************************
METHOD: DecisionTreeClassifier()
****************************************************************

Features: ['Pclass', 'Age', 'Deck', 'Sex_male', 'Num_relatives']
Standardized features: [None]
Tuned Model Parameters: {'criterion': 'gini', 'max_depth': 3}

========== Metrics ==========

Accuracy: 0.821
AUC: 0.802
R²: 0.821
Adjusted R²: 0.833

=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.84      0.88      0.86       111
         1.0       0.79      0.72      0.75        68

    accuracy                           0.82       179
   macro avg       0.81      0.80      0.81       179
weighted avg       0.82      0.82      0.82       179
```




```
<Figure size 504x288 with 0 Axes>
```

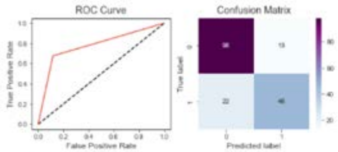


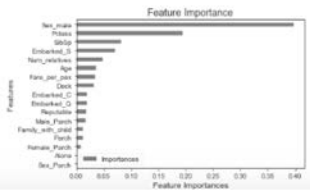
# XGBoost

## Full Model

```
METHOD: XGBClassifier()
************************************************************

Features: ['Pclass', 'Age', 'SibSp', 'Parch', 'Deck', 'Fare_per_pax', 'Sex_male', 'Embarked_C', 'Embarked_Q', '
Embarked_S', 'Reputable', 'Num_relatives', 'Alone', 'Family_with_child', 'Male_Parch', 'Female_Parch', 'Sex_Par
ch']
Standardized features: ['Pclass', 'Age', 'SibSp', 'Parch', 'Deck', 'Fare_per_pax', 'Sex_male', 'Embarked_C', 'E
mbarked_Q', 'Embarked_S', 'Reputable', 'Num_relatives', 'Alone', 'Family_with_child', 'Male_Parch', 'Female_Par
ch', 'Sex_Parch']
Tuned Model Parameters: {'xgb__colsample_bytree': 0.6, 'xgb__eta': 0.4, 'xgb__max_depth': 3, 'xgb__n_estimators
': 10, 'xgb__reg_alpha': 0.001, 'xgb__reg_lambda': 0.001}


========== Metrics ==========

Accuracy: 0.804
AUC: 0.780
R²: 0.804
Adjusted R²: 0.872


=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.82      0.88      0.85       111
         1.0       0.78      0.68      0.72        68

    accuracy                           0.80       179
   macro avg       0.80      0.78      0.79       179
weighted avg       0.80      0.80      0.80       179
```

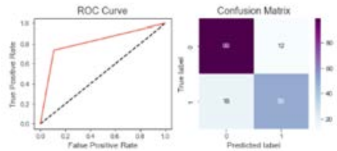



```
<Figure size 504x288 with 0 Axes>
```

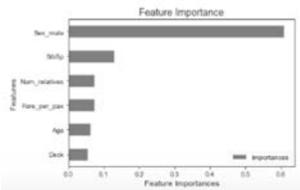



## Best Model

```
Fitting 3 folds for each of 972 candidates, totalling 2916 fits
************************************************************
METHOD: XGBClassifier()
************************************************************

Features: ['Age', 'SibSp', 'Fare_per_pax', 'Sex_male', 'Num_relatives', 'Deck']
Standardized features: ['Age', 'SibSp', 'Fare_per_pax', 'Sex_male', 'Num_relatives', 'Deck']
Tuned Model Parameters: {'xgb__colsample_bytree': 0.6, 'xgb__eta': 0.4, 'xgb__max_depth': 3, 'xgb__n_estimators
': 10, 'xgb__reg_alpha': 0.001, 'xgb__reg_lambda': 0.001}


========== Metrics ==========

Accuracy: 0.832
AUC: 0.814
R²: 0.832
Adjusted R²: 0.871


=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.85      0.89      0.87       111
         1.0       0.81      0.74      0.77        68

    accuracy                           0.83       179
   macro avg       0.83      0.81      0.82       179
weighted avg       0.83      0.83      0.83       179
```



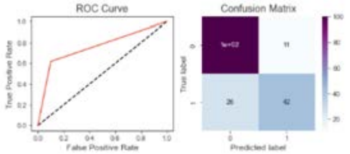

```
<Figure size 504x288 with 0 Axes>
```



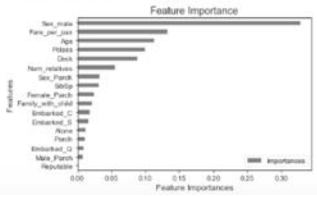
# Random Forest

## Full Model

```
*************************************************************
METHOD: RandomForestClassifier()
*************************************************************

Features: ['Pclass', 'Age', 'SibSp', 'Parch', 'Deck', 'Fare_per_pax', 'Sex_male', 'Embarked_C', 'Embarked_Q', '
Embarked_S', 'Reputable', 'Num_relatives', 'Alone', 'Family_with_child', 'Male_Parch', 'Female_Parch', 'Sex_Par
ch']
Standardized features: [None]
Tuned Model Parameters: {'criterion': 'entropy', 'max_depth': 5, 'n_estimators': 100}

========== Metrics ==========

Accuracy: 0.793
AUC: 0.759
R²: 0.793
Adjusted R²: 0.851

=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.79      0.90      0.84       111
         1.0       0.79      0.62      0.69        68

    accuracy                           0.79       179
   macro avg       0.79      0.76      0.77       179
weighted avg       0.79      0.79      0.79       179
```

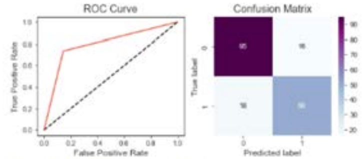



```
<Figure size 504x288 with 0 Axes>
```




## Best Model

```
*************************************************************
METHOD: RandomForestClassifier()
*************************************************************

Features: ['Age', 'SibSp', 'Fare_per_pax', 'Sex_male', 'Num_relatives', 'Deck']
Standardized features: [None]
Tuned Model Parameters: {'criterion': 'entropy', 'max_depth': 6, 'n_estimators': 100}

========== Metrics ==========

Accuracy: 0.810
AUC: 0.796
R²: 0.810
Adjusted R²: 0.860

=== Classification Report ===

              precision    recall  f1-score   support

         0.0       0.84      0.86      0.85       111
         1.0       0.76      0.74      0.75        68

    accuracy                           0.81       179
   macro avg       0.80      0.80      0.80       179
weighted avg       0.81      0.81      0.81       179
```




```
<Figure size 504x288 with 0 Axes>
```