# Exercise Set 4

Hina Arora

(I) Read data and pick relevant columns

(II) Split data into Train and Test

(III) Fit/Transform on Training Data

(IV) Transform/Predict on Test Data

## (I) Read data and pick relevant columns

- read data
- transform Cabin to Deck
- only retain columns required for analysis

# Question

Read kaggle train.csv data into a dataframe called df;
Then check the top few rows of the df dataframe.

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **531** | 532 | 0 | 3 | Toufik, Mr. Nakli | male | NaN | 0 | 0 | 2641 | 7.2292 | NaN | C |
| **305** | 306 | 1 | 1 | Allison, Master. Hudson Trevor | male | 0.92 | 1 | 2 | 113781 | 151.5500 | C22 C26 | S |
| **222** | 223 | 0 | 3 | Green, Mr. George Henry | male | 51.00 | 0 | 0 | 21440 | 8.0500 | NaN | S |

# Answer

```
df = pd.read_csv('data/kaggleTitanic/train.csv’)
df.sample(frac=0.01)
```

# Question

Extract a new column called “Deck” from “Cabin”.

Hint: write a function called getDeck, and then use following:
df['Deck'] = df['Cabin'].apply(getDeck)

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked | Deck |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **867** | 868 | 0 | 1 | Roebling, Mr. Washington Augustus II | male | 31.0 | 0 | 0 | PC 17590 | 50.4958 | A24 | S | A |
| **220** | 221 | 1 | 3 | Sunderland, Mr. Victor Francis | male | 16.0 | 0 | 0 | SOTON/OQ 392089 | 8.0500 | NaN | S | NaN |
| **361** | 362 | 0 | 2 | del Carlo, Mr. Sebastiano | male | 29.0 | 1 | 0 | SC/PARIS 2167 | 27.7208 | NaN | C | NaN |

# Answer

```
def getDeck(cabin):
    if pd.notna(cabin):
        return cabin[0]
    else:
        return np.nan

df['Deck'] = df['Cabin'].apply(getDeck)
df.sample(frac=0.01)
```

# (II) Split data into Train and Test

# Copy

```
# Set up data
X = df.drop(['Survived'], axis=1)
y = df['Survived’]

# Split into train and test
from sklearn.model_selection import train_test_split
Xtrain, Xtest, ytrain, ytest =
        train_test_split(X, y, test_size=0.2, random_state=1)

# Following code to deal with SetttingWithCopyWarning
Xtrain = Xtrain.copy()
Xtest = Xtest.copy()
ytrain = ytrain.copy()
ytest = ytest.copy()
```

# (III) Fit/Transform on Training Data

- 'Age’: impute missing values with median
- ['Pclass', 'Sex', 'Deck’]: impute missing values with ‘X’
- [‘impPclass', ‘impSex', 'impDeck']: OHE
- Only keep imputed numeric and ohe categorical features
- build Logistic Regression Model

# Question

Set numeric_features = ['Age’]

Use SimpleImputer to fill the missing values in numeric_features with the median values, and prefix the imputed columns with “imp”

| | PassengerId | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked | Deck | impAge |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **22** | 23 | 3 | McGowan, Miss. Anna "Annie" | female | 15.0 | 0 | 0 | 330923 | 8.0292 | NaN | Q | NaN | 15.0 |
| **15** | 16 | 2 | Hewlett, Mrs. (Mary D Kingcome) | female | 55.0 | 0 | 0 | 248706 | 16.0000 | NaN | S | NaN | 55.0 |
| **669** | 670 | 1 | Taylor, Mrs. Elmer Zebley (Juliet Cummins Wright) | female | NaN | 1 | 0 | 19996 | 52.0000 | C126 | S | C | 29.0 |

# Answer

```
numeric_features = ['Age']

from sklearn.impute import SimpleImputer
sinum = SimpleImputer(
        missing_values=np.nan,
        strategy='median')

Xnum = pd.DataFrame(
        sinum.fit_transform(Xtrain[numeric_features]),
        columns=['imp'+x for x in numeric_features],
        index=Xtrain.index)
Xtrain = pd.concat([Xtrain, Xnum], axis=1)

Xtrain.sample(7)
```

# Question

Set categorical_features = ['Pclass', 'Sex', 'Deck']

Use SimpleImputer to fill the missing values in categorical_features with the constant value ‘X’, and prefix the imputed columns with “imp”

| | PassengerId | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embarked | Deck | impAge | impPclass | impSex | impDeck |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **642** | 643 | 3 | Skoog, Miss. Margit Elizabeth | female | 2.0 | 3 | 2 | 347088 | 27.9000 | NaN | S | NaN | 2.0 | 3 | female | X |
| **183** | 184 | 2 | Becker, Master. Richard F | male | 1.0 | 2 | 1 | 230136 | 39.0000 | F4 | S | F | 1.0 | 2 | male | F |
| **465** | 466 | 3 | Goncalves, Mr. Manuel Estanslas | male | 38.0 | 0 | 0 | SOTON/O.Q. 3101306 | 7.0500 | NaN | S | NaN | 38.0 | 3 | male | X |

# Answer

```
categorical_features = ['Pclass', 'Sex', 'Deck']

from sklearn.impute import SimpleImputer
sicat = SimpleImputer(
        missing_values=np.nan,
        strategy='constant',
        fill_value='X')

Xcat = pd.DataFrame(
        sicat.fit_transform(Xtrain[categorical_features]),
        columns=['imp'+x for x in categorical_features],
        index=Xtrain.index)
Xtrain = pd.concat([Xtrain, Xcat], axis=1)

Xtrain.sample(7)
```

# Question

Set
imputed_categorical_features =
['impPclass', 'impSex', 'impDeck']

Use OneHotEncoder to one-hot-encode the imputed categorical variables

| | PassengerId | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | ... | x1_male | x2_A | x2_B | x2_C | x2_D | x2_E | x2_F | x2_G | x2_T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **670** | 671 | 2 | Brown, Mrs. Thomas William Solomon (Elizabeth ... | female | 40.0 | 1 | 1 | 29750 | 39.000 | NaN | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **526** | 527 | 2 | Ridsdale, Miss. Lucy | female | 50.0 | 0 | 0 | W./C. 14258 | 10.500 | NaN | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Answer

```
imputed_categorical_features =  ['impPclass', 'impSex', 'impDeck']

from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder(
        sparse=False,
        dtype=int,
        handle_unknown='ignore')

Xcat = pd.DataFrame(
        ohe.fit_transform(Xtrain[imputed_categorical_features]),
        columns=ohe.get_feature_names(),
        index=Xtrain.index)
Xtrain = pd.concat([Xtrain, Xcat], axis=1)

Xtrain.sample(7)
```

# Question

Only keep imputed numeric features, and ohe catergorical features

| | impAge | x0_1 | x0_2 | x0_3 | x1_female | x1_male | x2_A | x2_B | x2_C | x2_D | x2_E | x2_F | x2_G | x2_T | x2_X |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **45** | 29.0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **858** | 24.0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **438** | 64.0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **272** | 41.0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

# Answer

```
Xtrain.drop(
        ['PassengerId', 'Pclass', 'Name', 'Sex', 'Age',
                'SibSp', 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked',
                'Deck', 'impPclass', 'impSex', 'impDeck',],
        axis=1,
        inplace=True)

Xtrain.sample(7)
```

# Question

Build Logistic Regression Model by fitting to the transformed training data

# Answer

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(solver='liblinear')

lr.fit(Xtrain, ytrain)
```

# (IV) Transform/Predict on Test Data

- 'Age’: impute missing values with median
- ['Pclass', 'Sex', 'Deck’]: impute missing values with ‘X’
- [‘impPclass', ‘impSex', 'impDeck']: OHE
- Only keep imputed numeric and ohe categorical features
- predict using Logistic Regression Model
- evaluate model

# Question

numeric_features = ['Age’]

Fill the missing values in numeric_features with the median values

# Answer

```
Xnum = pd.DataFrame(
        sinum.transform(Xtest[numeric_features]),
        columns=['imp'+x for x in numeric_features],
        index=Xtest.index)
Xtest = pd.concat([Xtest, Xnum], axis=1)

Xtest.sample(7)
```

# Question

categorical_features = ['Pclass', 'Sex', 'Deck’]

Fill the missing values in categorical_features with the constant value ‘X’

# Answer

```
Xcat = pd.DataFrame(
        sicat.transform(Xtest[categorical_features]),
        columns=['imp'+x for x in categorical_features],
        index=Xtest.index)
Xtest = pd.concat([Xtest, Xcat], axis=1)

Xtest.sample(7)
```

# Question

imputed_categorical_features =
['impPclass', 'impSex', 'impDeck']

One-hot-encode the imputed categorical variables

# Answer

```
Xcat = pd.DataFrame(
        ohe.transform(Xtest[imputed_categorical_features]),
        columns=ohe.get_feature_names(),
        index=Xtest.index)
Xtest = pd.concat([Xtest, Xcat], axis=1)

Xtest.sample(7)
```

# Question

Only keep imputed numeric features, and ohe catergorical features

# Answer

```
Xtest.drop(
        ['PassengerId', 'Pclass', 'Name', 'Sex', 'Age',
                'SibSp', 'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked',
                'Deck', 'impPclass', 'impSex', 'impDeck',],
        axis=1,
        inplace=True)

Xtest.sample(7)
```

# Question

Predict and evaluate Logistic Regression Model on test data

# Answer

```
ypred = lr.predict(Xtest)

from sklearn import metrics
print (metrics.accuracy_score(ytest, ypred))
```

In this Exercise Set:

- We had to keep track of the preprocessing/transformation steps with "fit_transform" on the training data (including remembering to drop the columns we didn't want to use anymore)…. and then repeating all of the same preprocessing/transformation steps with "transform" on the test data

- We also had to keep track of building the model with “fit” on the preprocessed/transformed training data…. and then predicting with “predict” on the preprocessed/transformed test data

As we’ll see in the next exercise set, we can use ColumnTransformers and Pipelines to make life much-much easier for us!