In [ ]:

```
import pandas as pd
import numpy as np

from keras.models import Sequential
from keras.layers import Dense, Activation, BatchNormalization

from sklearn.model_selection import train_test_split
```

In [ ]:

```
df_train_org = pd.read_csv('train.csv')
df_train_rev = df_train_org.copy()

print(df_train_org.shape, df_train_rev.shape)
```

In [ ]:

```
df_train_org.head(10)
```

In [ ]:

```
# Sex
df_train_rev['Sex'] = df_train_org['Sex'].map( {'male': 0, 'female': 1} ).astype(int)

print(df_train_rev['Sex'].corr(df_train_rev['Survived']))
df_train_rev[["Sex", "Survived"]].groupby(['Sex'], as_index=False).mean()
```

In [ ]:

```
# Embarked
df_tmp = df_train_org.copy()
df_tmp['Embarked'] = df_train_org['Embarked'].fillna('S')
df_train_rev['Embarked'] = df_tmp['Embarked'].map( {'S': 0, 'Q': 0.5, 'C': 1} ).astype(int)

print(df_train_rev['Embarked'].corr(df_train_rev['Survived']))
df_train_rev[['Embarked', 'Survived']].groupby(['Embarked'], as_index=False).mean()
```

In [ ]:

```
# Pclass
df_train_rev['Pclass'] = df_train_org['Pclass'].map( {3: 0, 2: 0.5, 1: 1} ).astype(float)
df_train_rev[["Pclass", "Survived"]].groupby(['Pclass'], as_index=False).mean()
```

In [ ]:

```
# Fare
df_train_rev['Fare'] = (df_train_org['Fare'] - df_train_org['Fare'].mean()) / df_train_org['Far
e'].std()
df_train_rev['Fare'] = 1 / (1 + np.exp(-df_train_rev['Fare'])) # Sigmoid Function
df_train_rev['Fare'].describe()
```

In [ ]:

```
# Age
df_tmp['Age'] = df_train_org['Age'].fillna(df_train_org['Age'].mean())

df_train_rev['Age'] = (df_tmp['Age'] - df_train_org['Age'].mean()) / df_train_org['Age'].std()
df_train_rev['Age'] = 1 / (1 + np.exp(-df_train_rev['Age'])) # Sigmoid Function

print(df_train_rev['Age'].corr(df_train_rev['Survived']))
```

In [ ]:

```
# SibSp + Parch (FamilySize)
df_train_rev['FamilySize'] = (df_train_org['SibSp'] + df_train_org['Parch']) / (df_train_org['Si
bSp'].max() + df_train_org['Parch'].max())
```

In [ ]:

```
df_train_rev.head(10)
```

In [ ]:

```
df_train_rev.corrwith(df_train_rev['Survived'])
```

In [ ]:

```
# Merge
train_sex = df_train_rev['Sex'].values.reshape(-1, 1)
train_embarked = df_train_rev['Embarked'].values.reshape(-1, 1)
train_pclass = df_train_rev['Pclass'].values.reshape(-1, 1)
train_fare = df_train_rev['Fare'].values.reshape(-1, 1)
train_age = df_train_rev['Age'].values.reshape(-1, 1)
train_familysize = df_train_rev['FamilySize'].values.reshape(-1, 1)

data_input = np.concatenate((train_sex, train_embarked, train_pclass, train_fare, train_age, tra
in_familysize), axis = 1)
print(data_input.shape)
```

In [ ]:

```
data_label = df_train_rev['Survived'].values.reshape(-1, 1)
print(data_label.shape)
```

In [ ]:

```
h_units = 32
activation_ = 'relu'

def dnn_model():
    model = Sequential()

    model.add(Dense(units=h_units, input_dim=6))
    model.add(BatchNormalization())
    model.add(Activation(activation_))

    model.add(Dense(units=h_units))
    model.add(BatchNormalization())
    model.add(Activation(activation_))

    model.add(Dense(units=h_units))
    model.add(BatchNormalization())
    model.add(Activation(activation_))

    model.add(Dense(units=1))

    return model
```

In [ ]:

```
model = dnn_model()
model.compile(loss='mean_squared_error', optimizer='adagrad', metrics=['accuracy'])
```

In [ ]:

```
# split into training and validation datasets
x_train, x_val, y_train, y_val = train_test_split(data_input, data_label, test_size=0.2)
print(x_train.shape, y_train.shape, x_val.shape, y_val.shape)
```

In [ ]:

```
# training
model.fit(x_train, y_train, epochs=100, batch_size=16, verbose=True, validation_data=(x_val, y_v
al), shuffle=True)
```

In [ ]:

```
df_test_org = pd.read_csv('test.csv')
df_test_rev = df_test_org.copy()
print(df_test_org.shape, df_test_rev.shape)
```

In [ ]:

```
# Sex
df_test_rev['Sex'] = df_test_org['Sex'].map( {'male': 0, 'female': 1} ).astype(int)

# Embarked
df_tmp = df_test_org.copy()
df_tmp['Embarked'] = df_test_org['Embarked'].fillna('S')
df_test_rev['Embarked'] = df_tmp['Embarked'].map( {'S': 0, 'Q': 0.5, 'C': 1} ).astype(int)

# Pclass
df_test_rev['Pclass'] = df_test_org['Pclass'].map( {3: 0, 2: 0.5, 1: 1} ).astype(float)

# Fare
df_test_rev['Fare'] = (df_test_org['Fare'] - df_train_org['Fare'].mean()) / df_train_org['Fare']
.std()
df_test_rev['Fare'] = 1 / (1 + np.exp(-df_test_rev['Fare'])) # Sigmoid Function

# Age
df_test_rev['Age'] = df_test_org['Age'].fillna(df_train_org['Age'].mean())
df_test_rev['Age'] = (df_test_rev['Age'] - df_train_org['Age'].mean()) / df_train_org['Age'].std
()
df_test_rev['Age'] = 1 / (1 + np.exp(-df_test_rev['Age'])) # Sigmoid Function

# SibSp + Parch
df_test_rev['FamilySize'] = (df_test_rev['SibSp'] + df_test_rev['Parch']) / (df_train_org['SibS
p'].max() + df_train_org['Parch'].max())
```

In [ ]:

```
# Merge
test_sex = df_test_rev['Sex'].values.reshape(-1, 1)
test_embarked = df_test_rev['Embarked'].values.reshape(-1, 1)
test_pclass = df_test_rev['Pclass'].values.reshape(-1, 1)
test_fare = df_test_rev['Fare'].values.reshape(-1, 1)
test_age = df_test_rev['Age'].values.reshape(-1, 1)
test_familysize = df_test_rev['FamilySize'].values.reshape(-1, 1)

x_test = np.concatenate((test_sex, test_embarked, test_pclass, test_fare, test_age, test_familys
ize), axis = 1)
print(x_test.shape)
```

In [ ]:

```
y_pred = model.predict(x_test, batch_size=128)
y_pred = np.round(np.clip(y_pred, 0, 1))

print(y_pred.shape)
```

In [ ]:

```
submission = pd.DataFrame({
    "PassengerId" : df_test_org['PassengerId'].astype(int),
    "Survived" : y_pred.reshape(-1, ).astype(int)
})
```

In [ ]:

```
submission.head(10)
```

In [ ]:

```
submission.info()
```

In [ ]:

```
submission.to_csv('result.csv', index=False)
```