In [1]:

```
import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

%matplotlib inline
plt.style.use('ggplot')
```

In [2]:

```
df = pd.read_csv('Titanic.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Ca
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

In [4]:

```
def proc(str1):
    res = str1.split(',')[1].split('.')[0].strip()
    if res in ['Mr', 'Miss', 'Mrs', 'Master']:
        return res
    else:
        return 'Others'

df['Title']=df['Name'].astype(str).apply(proc)
```

In [5]:

```
df['HasCabin']=df['Cabin'].apply(pd.isnull)
```

In [6]:

```
df['Sex'].value_counts()
```

Out[6]:

```
male      577
female    314
Name: Sex, dtype: int64
```

In [7]:

```
df['Title'].value_counts()
```

Out[7]:

```
Mr        517
Miss      182
Mrs       125
Master     40
Others     27
Name: Title, dtype: int64
```

In [8]:

```
df['Embarked'].value_counts()
```

Out[8]:

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [9]:

```
for t in ['Sex', 'Title', 'Embarked']:
    df = pd.concat([df, pd.get_dummies(df[t])], axis=1)
```

In [10]:

```
df.head()
```

Out[10]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	...
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	...
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	...
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	...
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	...
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	...

5 rows × 24 columns

In [11]:

```
df.columns
```

Out[11]:

```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'Sib
Sp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked', 'Title', 'HasCa
bin',
       'female', 'male', 'Master', 'Miss', 'Mr', 'Mrs', 'Others', 'C',
'Q',
       'S'],
      dtype='object')
```

In [12]:

```
feats = [t for t in df.columns
         if t not in ['Survived', 'Name', 'Ticket', 'Cabin', 'Sex', 'Title', 'Embark
target = ['Survived']
```

In [13]:

```
df_train, df_test = train_test_split(df, stratify=df['Survived'], shuffle=True, test
```

In [14]:

```
data_trn = xgb.DMatrix(df_train[feats], label=df_train[target])
data_val = xgb.DMatrix(df_test[feats], label=df_test[target])
```

In [60]:

```
param = {
    'silent': 1,
    'eta': 0.08,
    'gamma': 0,
    'max_depth ': 2,
    'min_child_weight': 1,
    'subsample': 1,
    'lambda': 1,
    'alpha': 0,
    'bjective': 'binary:logistic',
    'eval_metric': 'logloss',
}
```

In [61]:

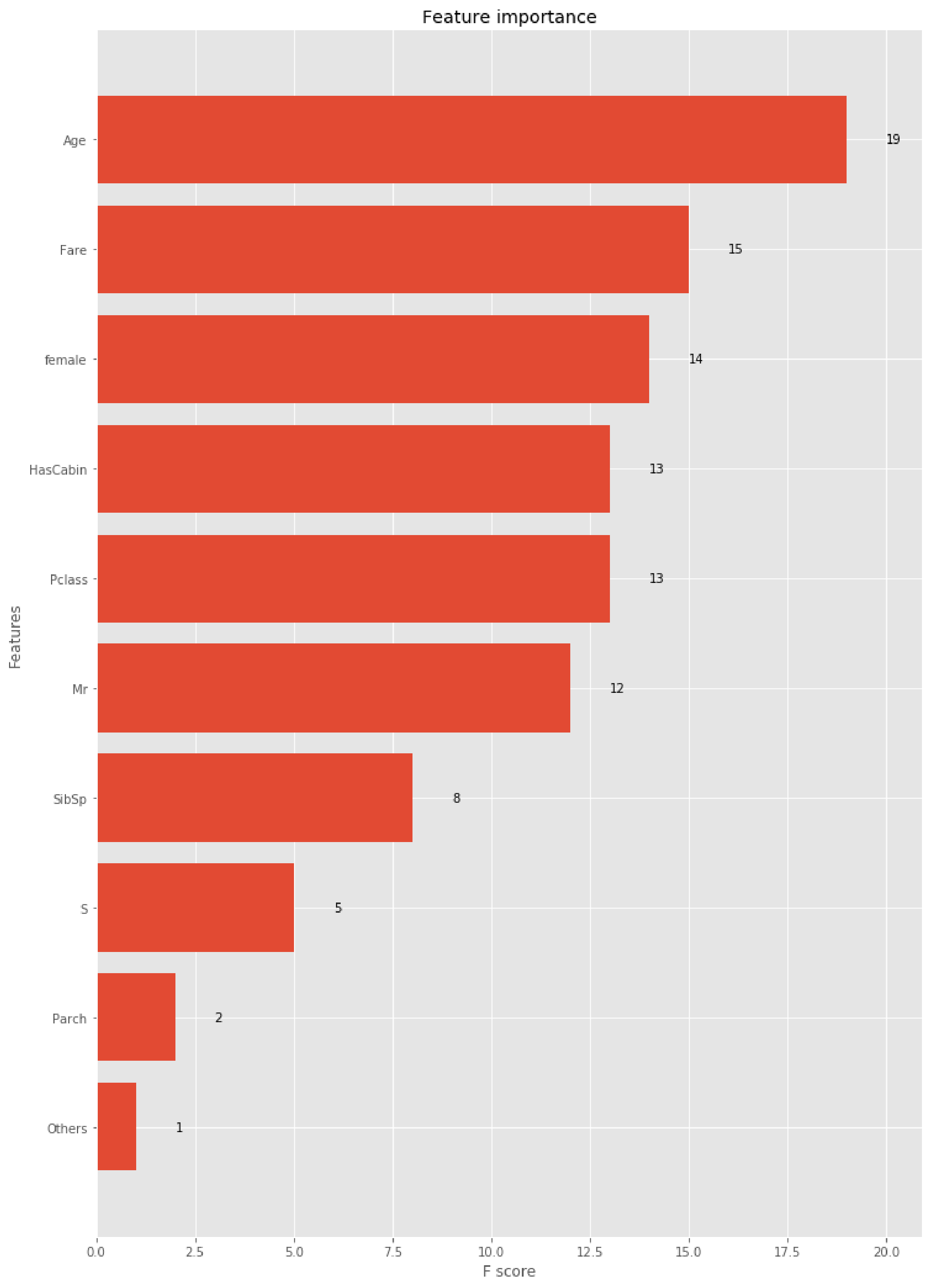
```
evals_result = {}
bst = xgb.train(
    params=param, # Booster params
    dtrain=data_trn, # Data to be trained
    num_boost_round=200, #  Number of boosting iterations
    evals=[(data_trn, 'train'), (data_val, 'eval')], # List of items to be evaluate
    obj=None, # Customized objective function
    feval=None, # Customized evaluation function
    maximize=False, # Whether to maximize feval
    early_stopping_rounds=3, # Validation error needs to decrease at least every <ea
    evals_result=evals_result, # This dictionary stores the evaluation results of a
    verbose_eval=2,
    learning_rates=None, # List of learning rate for each boosting round
    xgb_model=None,
    callbacks=None, # list of callback functions
)
```

```
[0]     train-logloss:0.644084  eval-logloss:0.655657
Multiple eval metrics have been passed: 'eval-logloss' will be used fo
r early stopping.

Will train until eval-logloss hasn't improved in 3 rounds.
[2]     train-logloss:0.563202  eval-logloss:0.597299
[4]     train-logloss:0.500736  eval-logloss:0.558651
[6]     train-logloss:0.449635  eval-logloss:0.529011
[8]     train-logloss:0.407089  eval-logloss:0.507065
[10]    train-logloss:0.37451   eval-logloss:0.492354
[12]    train-logloss:0.344772  eval-logloss:0.481165
[14]    train-logloss:0.320689  eval-logloss:0.471064
[16]    train-logloss:0.298508  eval-logloss:0.462935
[18]    train-logloss:0.280579  eval-logloss:0.459645
[20]    train-logloss:0.263187  eval-logloss:0.458922
[22]    train-logloss:0.249087  eval-logloss:0.458764
Stopping. Best iteration:
[19]    train-logloss:0.271463  eval-logloss:0.458588
```

In [56]:

```
fig, ax = plt.subplots(figsize=(12,18))
xgb.plot_importance(bst, max_num_features=50, height=0.8, ax=ax, importance_type='we
plt.show()
```

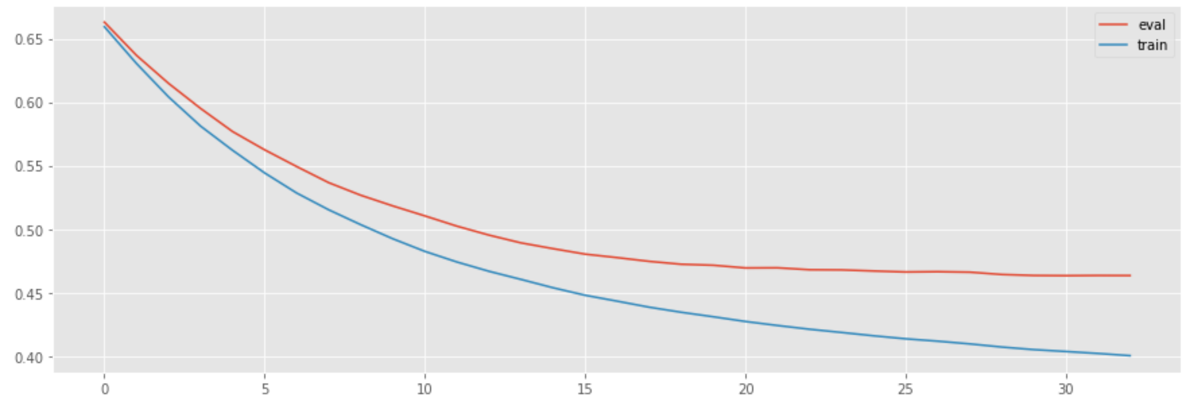


In [57]:

```
pd.DataFrame({'eval': evals_result['eval']['logloss'], 'train': evals_result['train
```

Out[57]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a1d4b4f28>
```



In [ ]:

```

```