In [1]:

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/titanic/train.csv
/kaggle/input/titanic/test.csv
/kaggle/input/titanic/gender_submission.csv
```

In [2]:

```
train_df = pd.read_csv('/kaggle/input/titanic/train.csv')
train_df.head()
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [3]:

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [4]:

```
train_df.describe()
```

Out[4]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [5]:

```
test_df = pd.read_csv('/kaggle/input/titanic/test.csv')
test_df.head()
```

Out[5]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

In [6]:

```
gender_submission_df = pd.read_csv('/kaggle/input/titanic/gender_submission.csv')
gender_submission_df.head()
```

Out[6]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1

In [7]:

```
train_df["Sex"].value_counts()
```

Out[7]:

```
male      577
```

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

In [8]:

```
train_df["Survived"].value_counts()
```

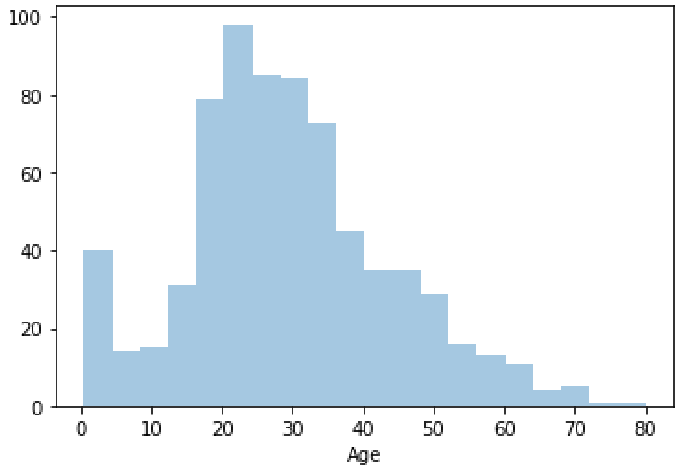
Out[8]:

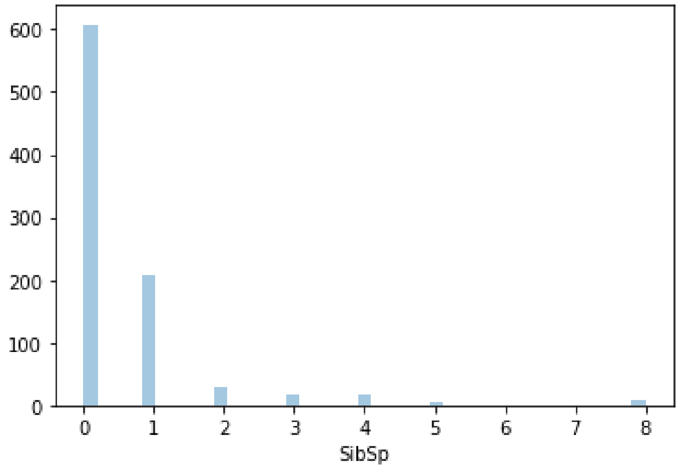
```
0    549
1    342
Name: Survived, dtype: int64
```

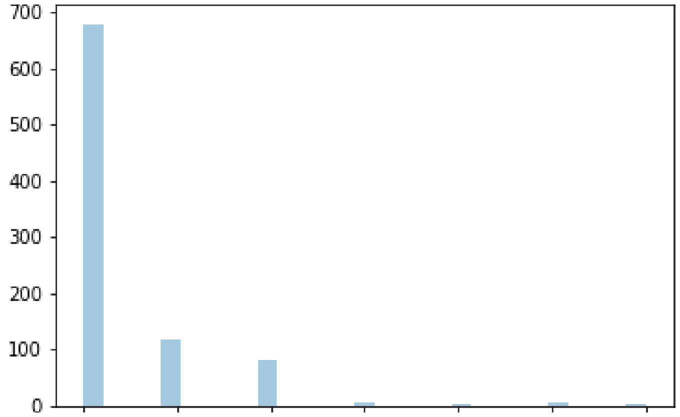
In [9]:

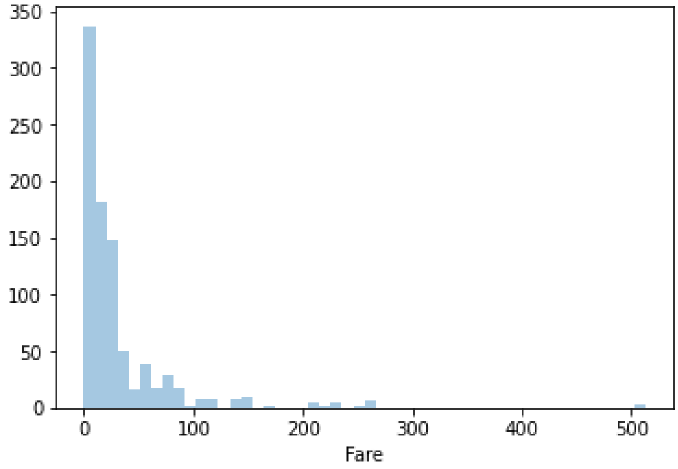
```
df_num = train_df[['Age', 'SibSp', 'Parch', 'Fare']]
for i in df_num.columns:
    sns.distplot(a=train_df[i], kde=False)
    plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2557: FutureWarning: `dis
tplot` is a deprecated function and will be removed in a future version. Please adapt you
r code to use either `displot` (a figure-level function with similar flexibility) or `his
tplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```





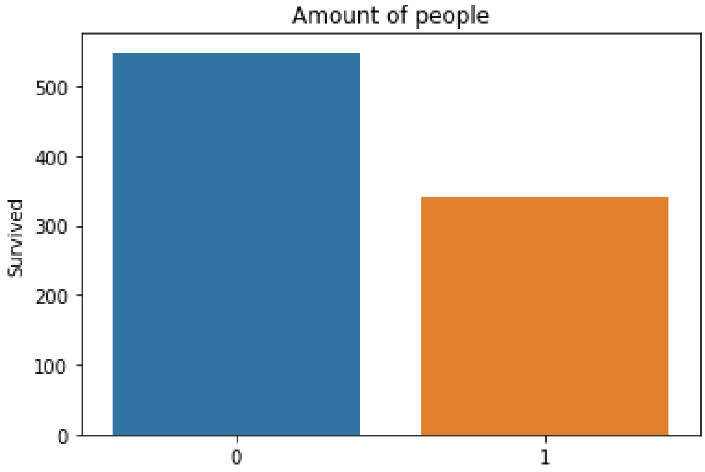




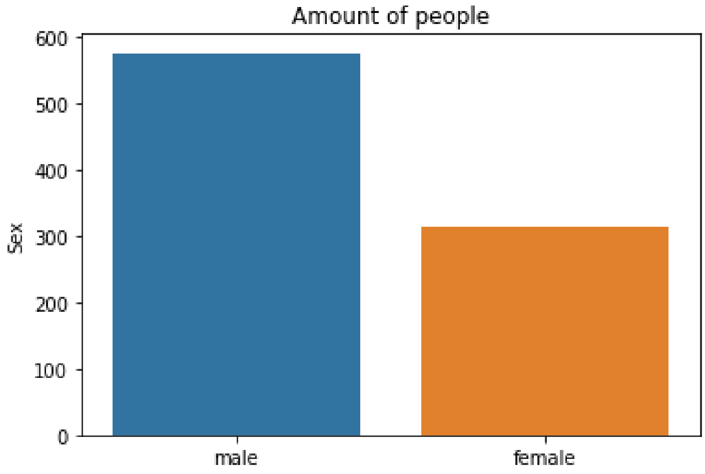
In [10]:

```
df_categories = train_df[['Survived', 'Sex', 'Embarked', 'Pclass']]
for i in df_categories.columns:
    sns.barplot(df_categories[i].value_counts().index,df_categories[i].value_counts()).s
et_title("Amount of people")
    plt.show()
```

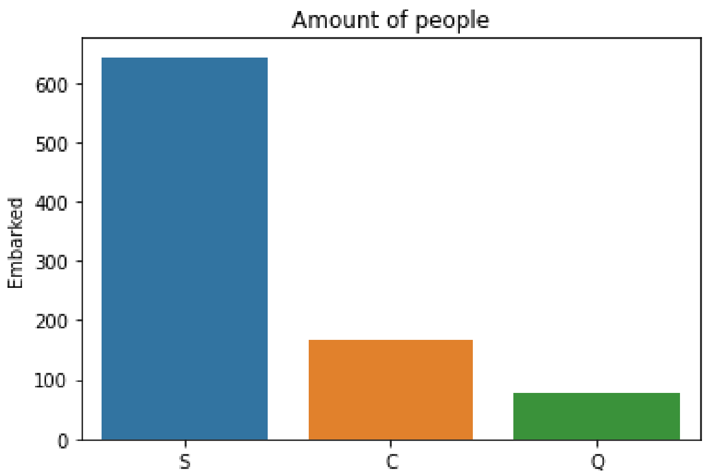
```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will resu
lt in an error or misinterpretation.
  FutureWarning
```



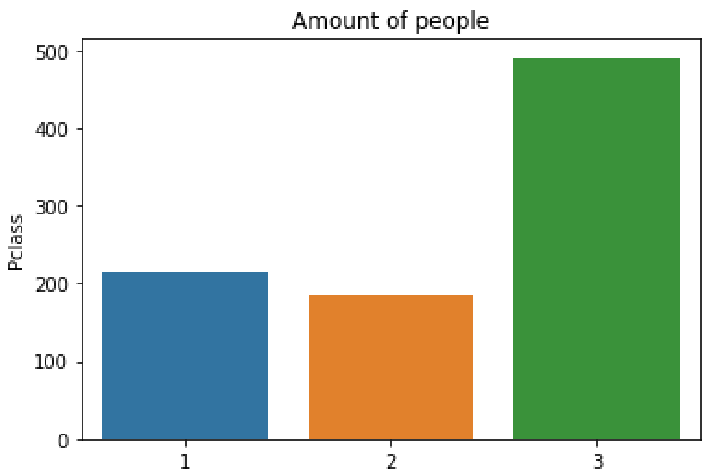
```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will resu
lt in an error or misinterpretation.
  FutureWarning
```



```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



```
/opt/conda/lib/python3.7/site-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

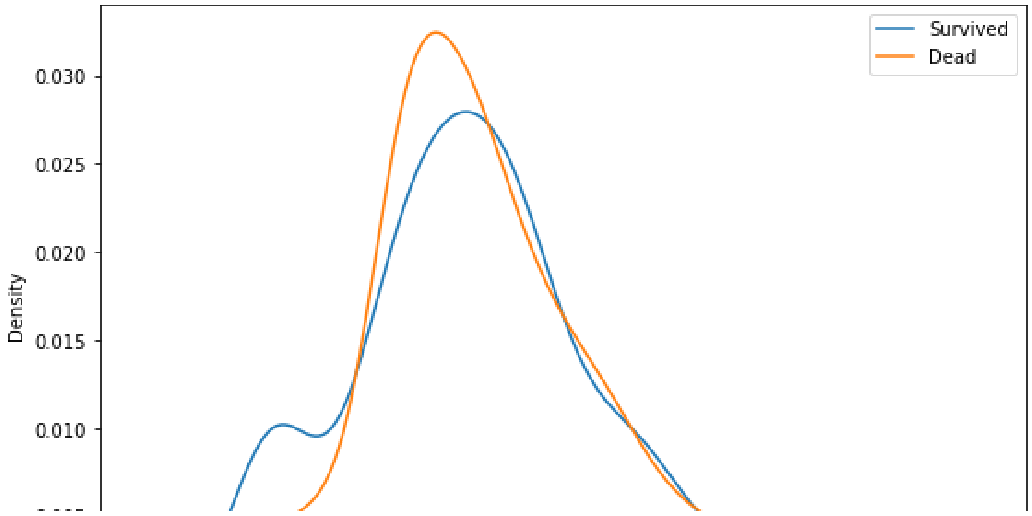


In [11]:

```
plt.figure(figsize=(9, 6))
sns.kdeplot(train_df[train_df['Survived'] == 1]['Age'])
sns.kdeplot(train_df[train_df['Survived'] == 0]['Age'])
plt.legend(['Survived', 'Dead'])
```

Out[11]:

```
<matplotlib.legend.Legend at 0x7f38f418d790>
```

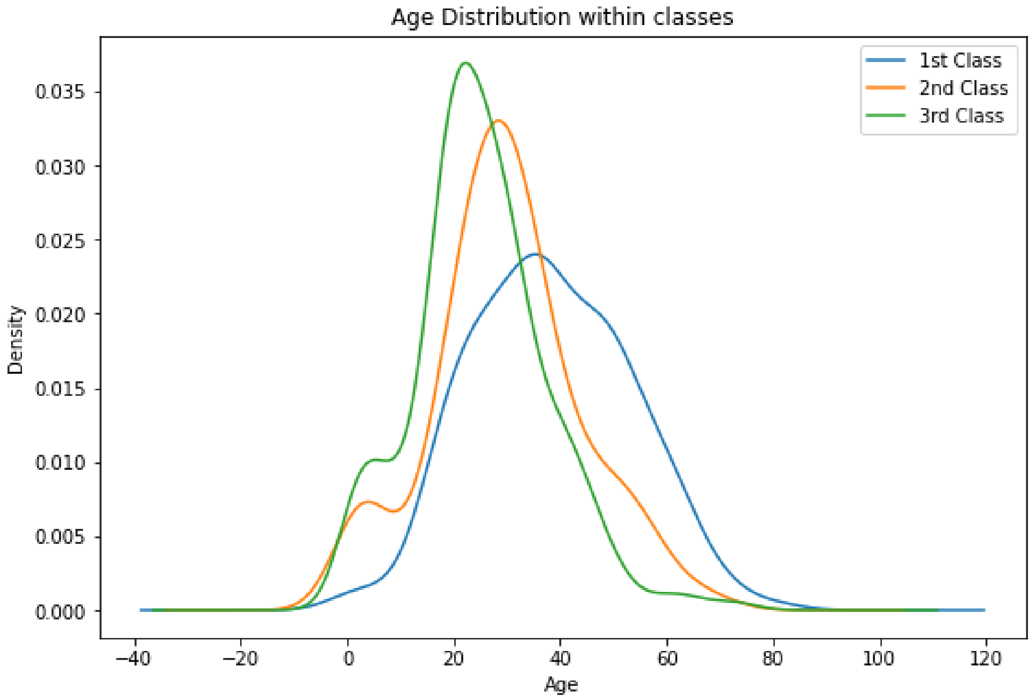


In [12]:

```
plt.figure(figsize=(9, 6))
for i in range(1,4):
    train_df['Age'][train_df['Pclass'] == i].plot(kind='kde')
plt.xlabel('Age')
plt.title('Age Distribution within classes')
plt.legend(['1st Class', '2nd Class', '3rd Class'])
```

Out[12]:

```
<matplotlib.legend.Legend at 0x7f38f00a5750>
```



In [13]:

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error

from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeRegressor
```

In [14]:

```
#get mean absolute error according to the number of leaf_nodes
def get_mae(max_leaf_nodes, train_X, val_X, train_y, val_y):
    model = DecisionTreeRegressor(max_leaf_nodes=max_leaf_nodes, random_state=0)
    model.fit(train_X, train_y)
    preds_val = model.predict(val_X)
    mae = mean_absolute_error(val_y, preds_val)
    return(mae)
```

In [15]:

```
#target
y = train_df.Survived

features = ["Pclass", "Sex", "Parch"]
X = pd.get_dummies(train_df[features])
X_test = pd.get_dummies(test_df[features])
```

In [16]:

```
model = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=1)
model.fit(X, y)
predictions = model.predict(X_test)

model_2=DecisionTreeRegressor(max_leaf_nodes=100,random_state=1)
model_2.fit(X, y)
predictions_2 = model_2.predict(X_test)

output = pd.DataFrame({'PassengerId': test_df.PassengerId, 'Survived': predictions})
output.to_csv('my_submission_1.csv', index=False)
output = pd.DataFrame({'PassengerId': test_df.PassengerId, 'Survived': predictions_2})
output.to_csv('my_submission_2.csv', index=False)

print("Your submission was successfully saved!")
```

```
Your submission was successfully saved!
```

In [17]:

```
#APART:We are going to compare two different machine learning model



# Split into validation and training data
train_X, val_X, train_y, val_y = train_test_split(X, y, test_size = 0.2, random_state=1)


candidate_max_leaf_nodes = [5, 25, 50, 100, 250, 500, 1000]
# Write loop to find the ideal tree size from candidate_max_leaf_nodes
for leaf in candidate_max_leaf_nodes:
    mae1 = get_mae(leaf, train_X, val_X, train_y, val_y)
    if leaf == 5:
        mae2 = mae1
        score = leaf
    if mae1 < mae2:
        mae2 = mae1
        score = leaf
# Store the best value of max_leaf_nodes (it will be either 5, 25, 50, 100, 250 or 500, 1
000)
best_tree_size = score


# Specify Model
dt_model = DecisionTreeRegressor(max_leaf_nodes = best_tree_size, random_state=1)
rf_model = RandomForestRegressor(random_state=1)
# Fit Model
dt_model.fit(train_X, train_y)
rf_model.fit(train_X, train_y)

# Make validation predictions and calculate mean absolute error
dt_predictions = dt_model.predict(val_X)
rf_predictions = rf_model.predict(val_X)

dt_mae = mean_absolute_error(val_y, dt_predictions)
rf_mae = mean_absolute_error(val_y, rf_predictions)

#display
print("DT MAE: "+str(dt_mae))
print("RF MAE: "+ str(rf_mae))
```

```
DT MAE: 0.3061227180429256
RF MAE: 0.30525345361142464
```