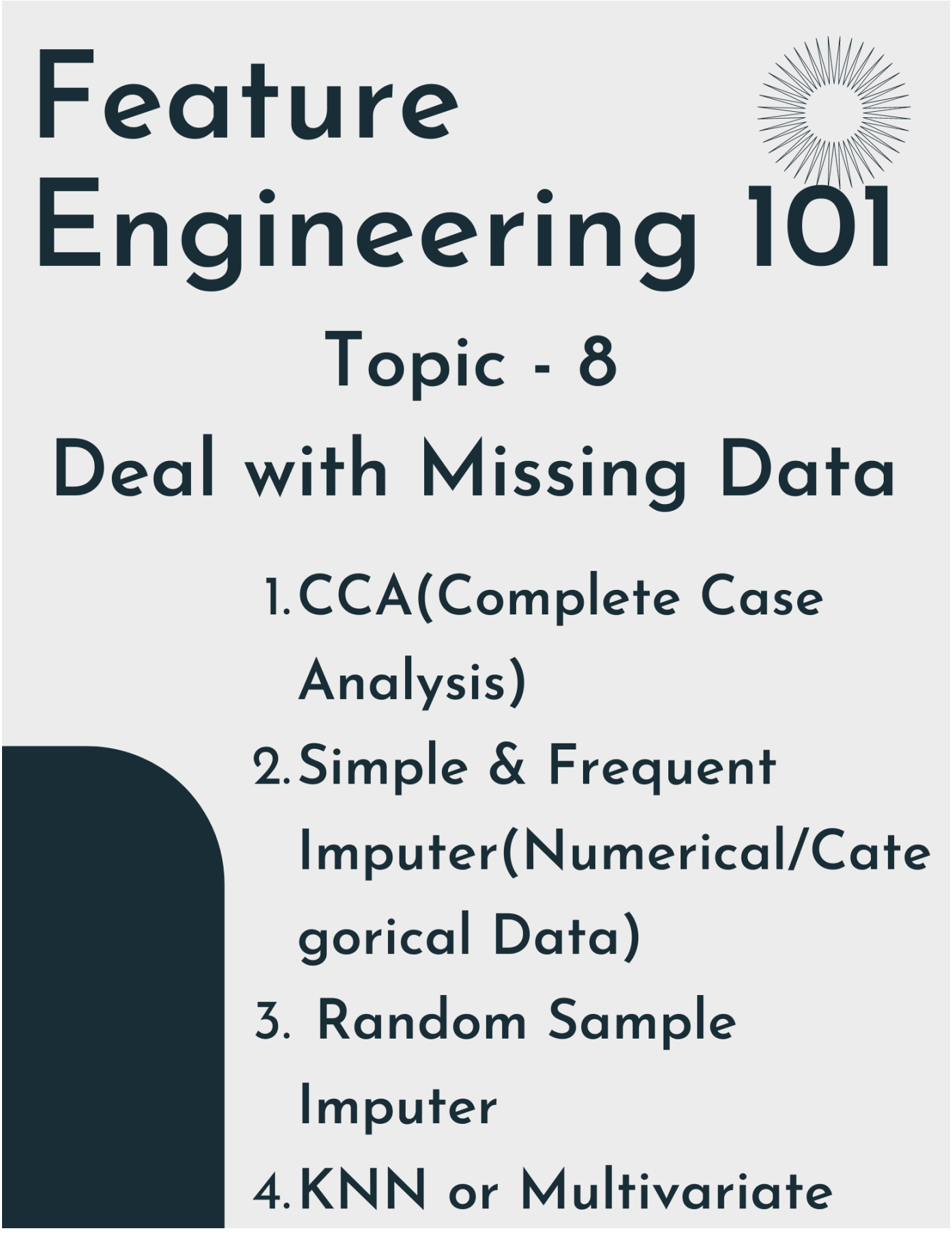


CCA (Complete Case Analysis)

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv('data_science_job.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	enrollee_id	city	city_development_index	gender	relevent_experience	enrolled_university	education_level	m
0	8949	city_103	0.920	Male	Has relevent experience	no_enrollment	Graduate	
1	29725	city_40	0.776	Male	No relevent experience	no_enrollment	Graduate	
2	11561	city_21	0.624	NaN	No relevent experience	Full time course	Graduate	
3	33241	city_115	0.789	NaN	No relevent experience	NaN	Graduate	B
4	666	city_162	0.767	Male	Has relevent experience	no_enrollment	Masters	

In [4]:
```
df.isnull().mean()*100
```

Out[4]:
```
enrollee_id              0.000000
city                     0.000000
city_development_index   2.500261
gender                  23.530640
relevent_experience      0.000000
enrolled_university      2.014824
education_level          2.401086
major_discipline        14.683161
experience               0.339284
company_size            30.994885
company_type            32.049274
training_hours           3.998330
target                   0.000000
dtype: float64
```

In [5]:
```
df.shape
```

Out[5]:
```
(19158, 13)
```

In [6]:
```
cols = [var for var in df.columns if df[var].isnull().mean() < 0.05 and df[var].isnull().m
cols
```

Out[6]:
```
['city_development_index',
 'enrolled_university',
 'education_level',
 'experience',
 'training_hours']
```

In [7]:
```
df[cols].sample(5)
```

Out[7]:

	city_development_index	enrolled_university	education_level	experience	training_hours
2179	0.899	no_enrollment	Graduate	8.0	54.0
11724	0.884	no_enrollment	Graduate	20.0	NaN

	city_development_index	enrolled_university	education_level	experience	training_hours
7189	0.897	no_enrollment	Masters	16.0	156.0
748	0.920	no_enrollment	Graduate	20.0	55.0
13129	0.920	no_enrollment	Graduate	20.0	3.0

In [8]:
```
df['education_level'].value_counts()
```

Out[8]:
```
Graduate          11598
Masters            4361
High School        2017
Phd                 414
Primary School      308
Name: education_level, dtype: int64
```

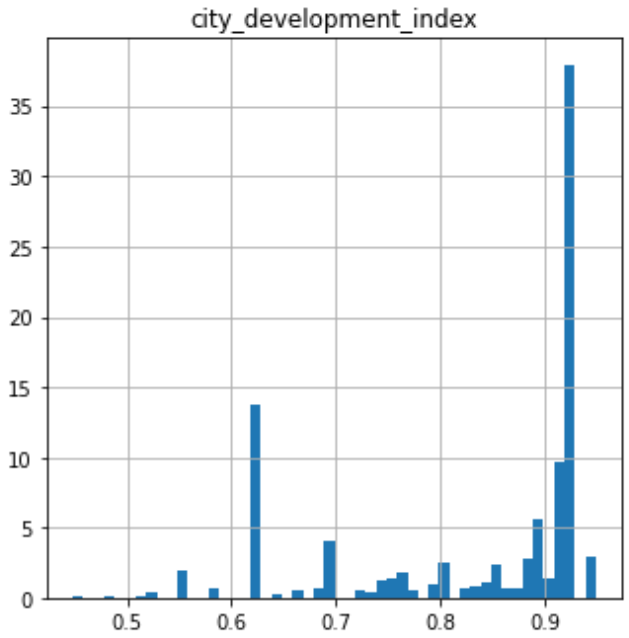
In [9]:
```
len(df[cols].dropna()) / len(df)
```

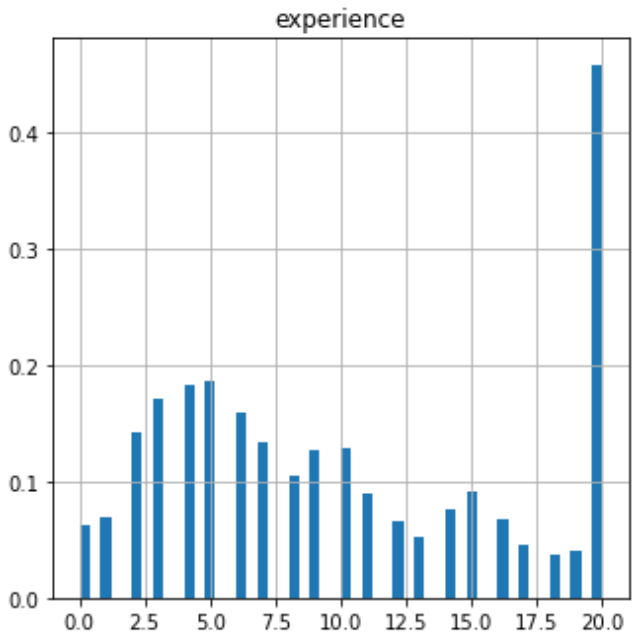
Out[9]:
```
0.8968577095730244
```

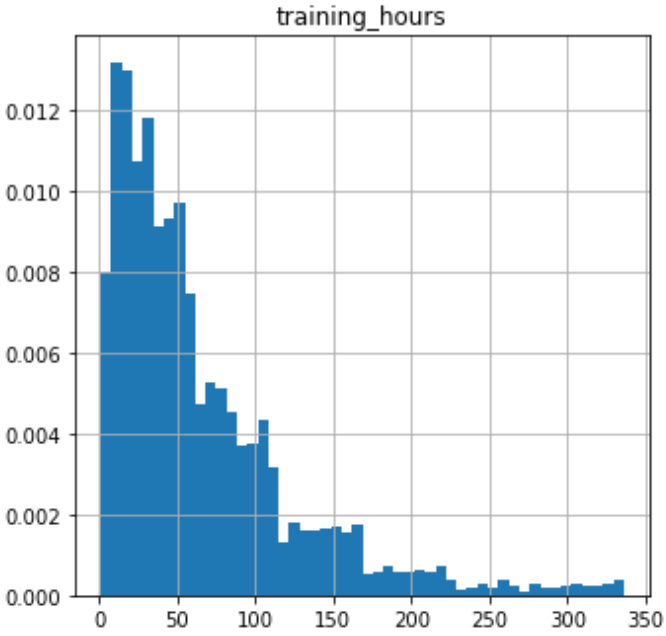
In [10]:
```
new_df = df[cols].dropna()
df.shape, new_df.shape
```

Out[10]:
```
((19158, 13), (17182, 5))
```

In [11]:
```
new_df.hist(bins=50, density=True, figsize=(12, 12))
plt.show()
```

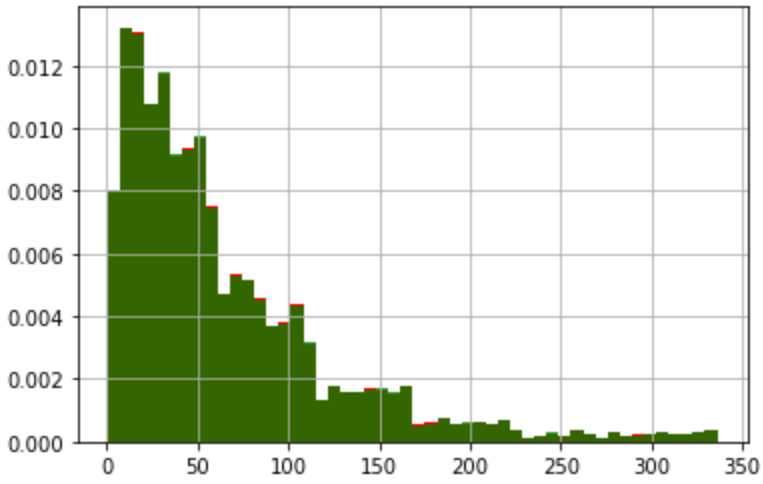






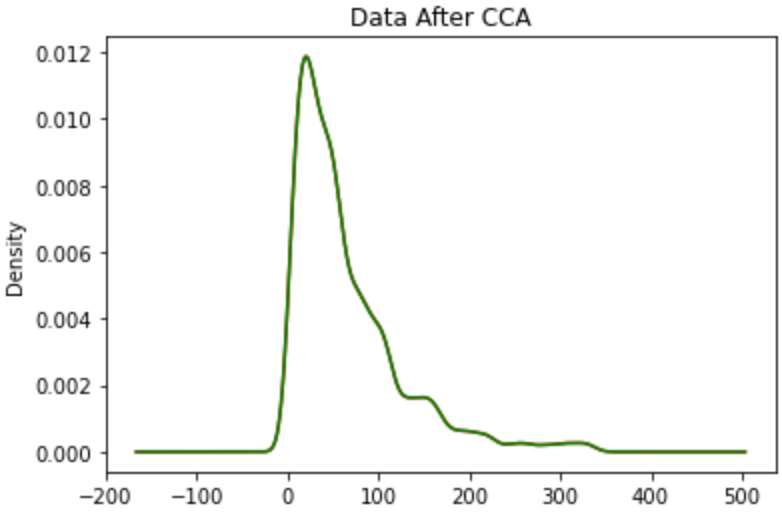
In [12]:

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['training_hours'].hist(bins=50, ax=ax, density=True, color='red')

# data after cca, the argument alpha makes the color transparent, so we can
# see the overlay of the 2 distributions
new_df['training_hours'].hist(bins=50, ax=ax, color='green', density=True, alpha=0.8)
plt.show()
```



In [13]:

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['training_hours'].plot.density(color='red')

# Data After CCA
new_df['training_hours'].plot.density(color='green')
plt.title('Data After CCA')
plt.show()
```

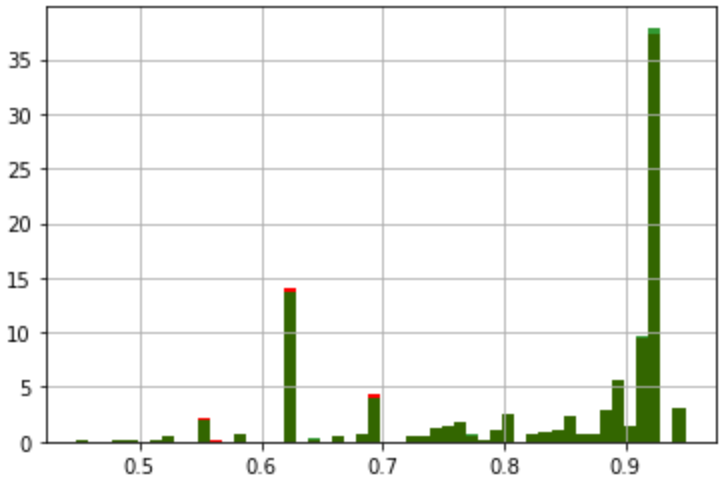


In [14]:

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['city_development_index'].hist(bins=50, ax=ax, density=True, color='red')

# data after cca, the argument alpha makes the color transparent, so we can
# see the overlay of the 2 distributions
new_df['city_development_index'].hist(bins=50, ax=ax, color='green', density=True, alpha=0
```

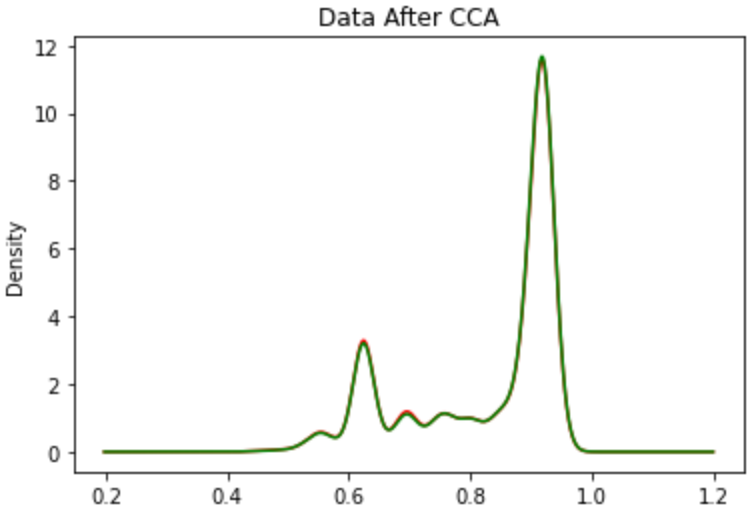
Out[14]:

```
<AxesSubplot:>
```



In [15]:

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['city_development_index'].plot.density(color='red')

# data after cca
new_df['city_development_index'].plot.density(color='green')
plt.title('Data After CCA')
plt.show()
```

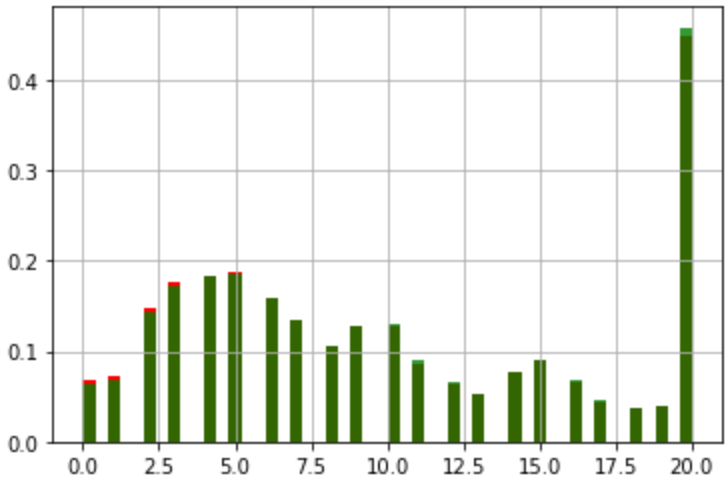


In [16]:

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['experience'].hist(bins=50, ax=ax, density=True, color='red')

# data after cca, the argument alpha makes the color transparent, so we can
# see the overlay of the 2 distributions
new_df['experience'].hist(bins=50, ax=ax, color='green', density=True, alpha=0.8)
```

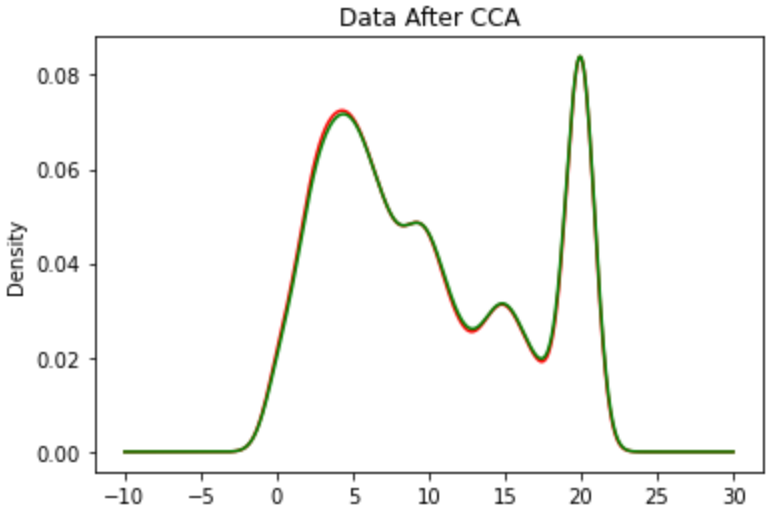
Out[16]:

```
<AxesSubplot:>
```



In [17]:

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original data
df['experience'].plot.density(color='red')

# data after cca
new_df['experience'].plot.density(color='green')
plt.title('Data After CCA')
plt.show()
```



In [18]:

```
temp = pd.concat([
            # percentage of observations per category, original data
            df['enrolled_university'].value_counts() / len(df),

            # percentage of observations per category, cca data
            new_df['enrolled_university'].value_counts() / len(new_df)
        ],
        axis=1)

# add column names
temp.columns = ['original', 'cca']

temp
```

Out[18]:

	original	cca
no_enrollment	0.721213	0.735188

	original	cca
Full time course	0.196106	0.200733
Part time course	0.062533	0.064079

In [19]:

```
temp = pd.concat([
            # percentage of observations per category, original data
            df['education_level'].value_counts() / len(df),

            # percentage of observations per category, cca data
            new_df['education_level'].value_counts() / len(new_df)
        ],
        axis=1)

# add column names
temp.columns = ['original', 'cca']

temp
```

Out[19]:

	original	cca
Graduate	0.605387	0.619835
Masters	0.227633	0.234082
High School	0.105282	0.107380
Phd	0.021610	0.022116
Primary School	0.016077	0.016587

In [ ]:

Handling Missing Categorical Data (frequent-value-imputation)

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
df = pd.read_csv('train1.csv',usecols=['GarageQual','FireplaceQu','SalePrice'])
```

In [3]:
```
df.head()
```

Out[3]:

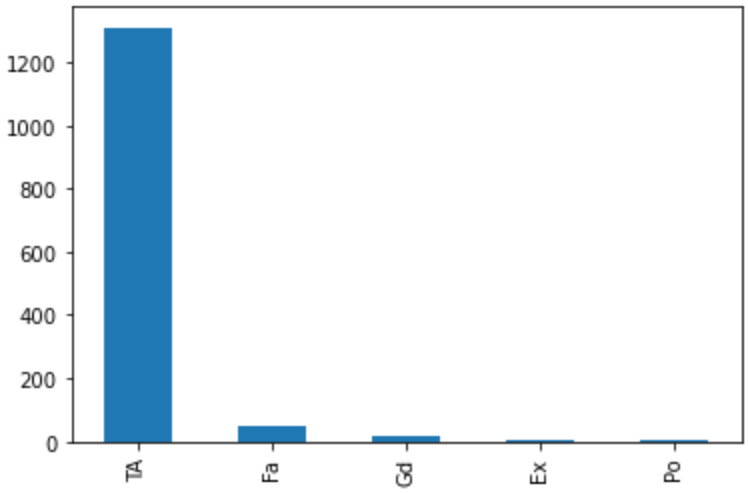
	FireplaceQu	GarageQual	SalePrice
0	NaN	TA	208500
1	TA	TA	181500
2	TA	TA	223500
3	Gd	TA	140000
4	TA	TA	250000

In [4]:
```
df.isnull().mean()*100
```

Out[4]:
```
FireplaceQu    47.260274
GarageQual      5.547945
SalePrice       0.000000
dtype: float64
```

In [5]:
```
df['GarageQual'].value_counts().plot(kind='bar')
```

Out[5]:
```
<AxesSubplot:>
```

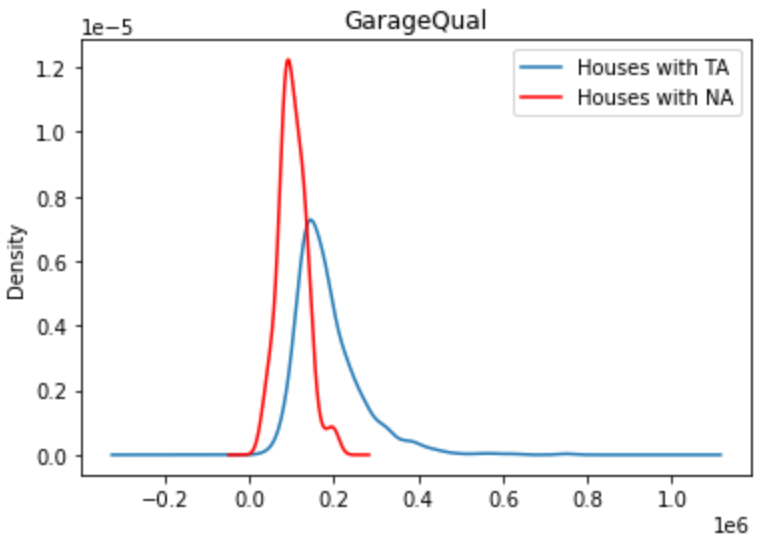


In [6]:
```
df['GarageQual'].mode()
```

Out[6]:
```
0    TA
dtype: object
```

In [7]:
```
fig = plt.figure()
ax = fig.add_subplot(111)

df[df['GarageQual']=='TA']['SalePrice'].plot(kind='kde', ax=ax)

df[df['GarageQual'].isnull()]['SalePrice'].plot(kind='kde', ax=ax, color='red')

lines, labels = ax.get_legend_handles_labels()
labels = ['Houses with TA', 'Houses with NA']
ax.legend(lines, labels, loc='best')

plt.title('GarageQual')
```

Out[7]:
```
Text(0.5, 1.0, 'GarageQual')
```

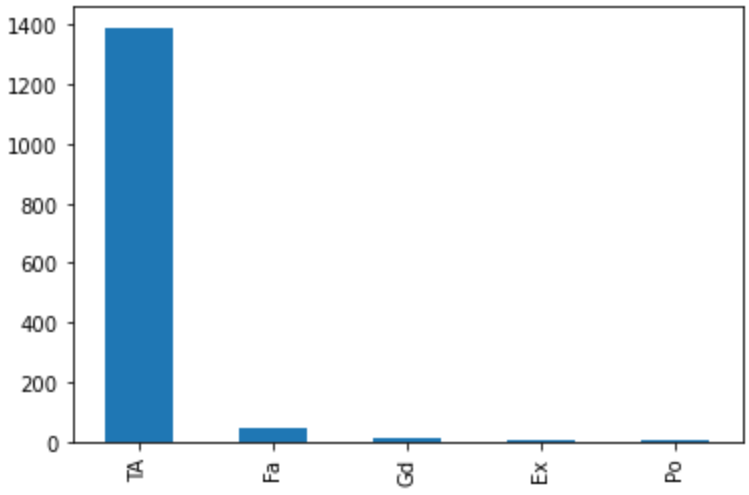


In [8]:
```
temp = df[df['GarageQual']=='TA']['SalePrice']
```

In [9]:
```
df['GarageQual'].fillna('TA', inplace=True)
```

In [10]:
```
df['GarageQual'].value_counts().plot(kind='bar')
```

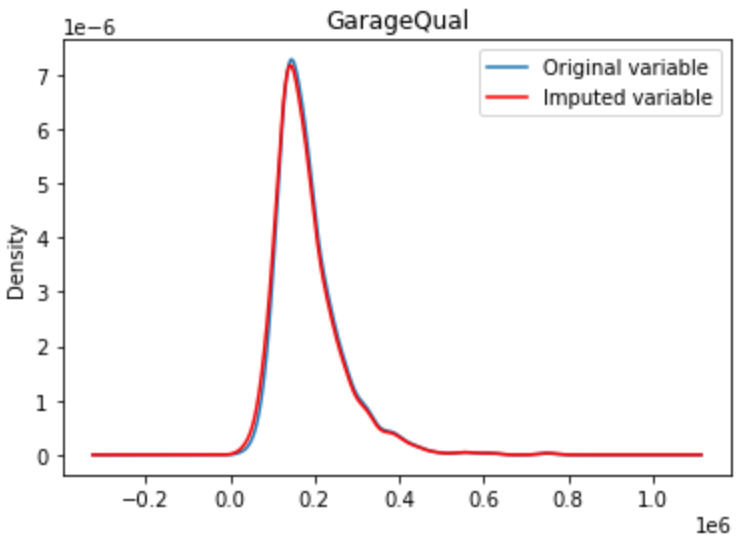
Out[10]:
```
<AxesSubplot:>
```



In [11]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
```

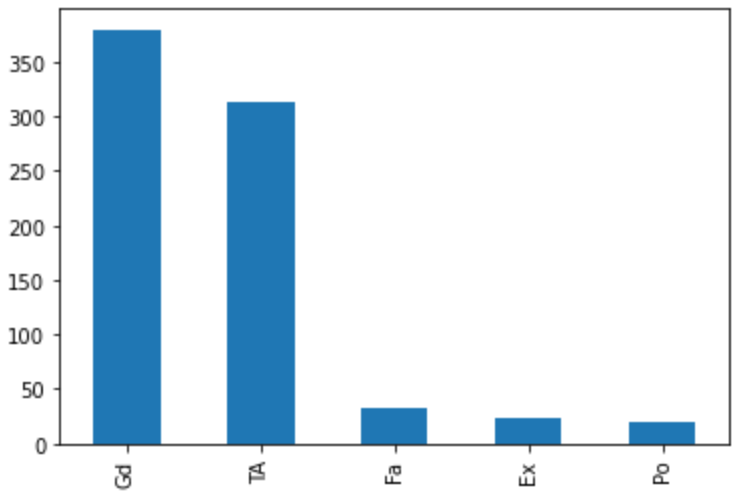
```
temp.plot(kind='kde', ax=ax)

# distribution of the variable after imputation
df[df['GarageQual'] == 'TA']['SalePrice'].plot(kind='kde', ax=ax, color='red')

lines, labels = ax.get_legend_handles_labels()
labels = ['Original variable', 'Imputed variable']
ax.legend(lines, labels, loc='best')

# add title
plt.title('GarageQual')
```

Out[11]:
```
Text(0.5, 1.0, 'GarageQual')
```



In [12]:
```
df['FireplaceQu'].value_counts().plot(kind='bar')
```

Out[12]:
```
<AxesSubplot:>
```



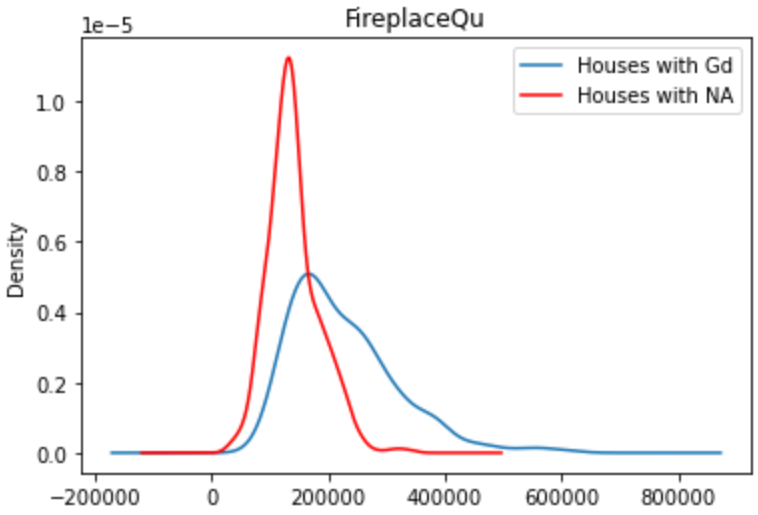
In [13]:
```
df['FireplaceQu'].mode()
```

Out[13]:
```
0    Gd
dtype: object
```

In [14]:
```
fig = plt.figure()
ax = fig.add_subplot(111)

df[df['FireplaceQu']=='Gd']['SalePrice'].plot(kind='kde', ax=ax)
```

```
df[df['FireplaceQu'].isnull()]['SalePrice'].plot(kind='kde', ax=ax, color='red')

lines, labels = ax.get_legend_handles_labels()
labels = ['Houses with Gd', 'Houses with NA']
ax.legend(lines, labels, loc='best')

plt.title('FireplaceQu')
```

Out[14]:
```
Text(0.5, 1.0, 'FireplaceQu')
```

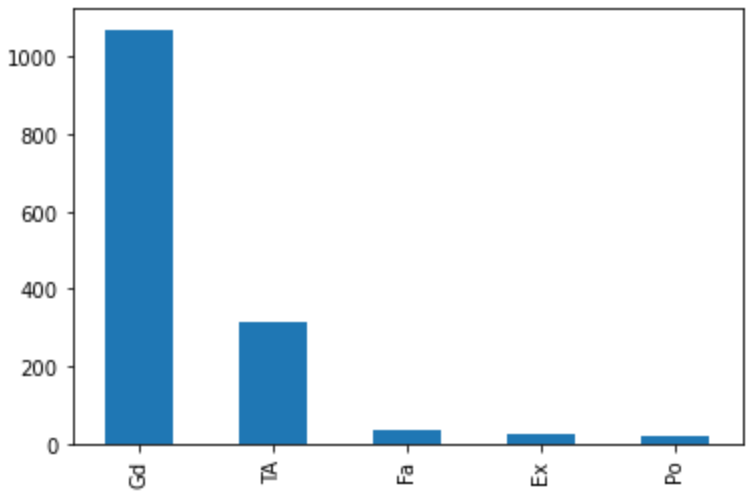


In [15]:
```
temp = df[df['FireplaceQu']=='Gd']['SalePrice']
```

In [16]:
```
df['FireplaceQu'].fillna('Gd', inplace=True)
```

In [17]:
```
df['FireplaceQu'].value_counts().plot(kind='bar')
```

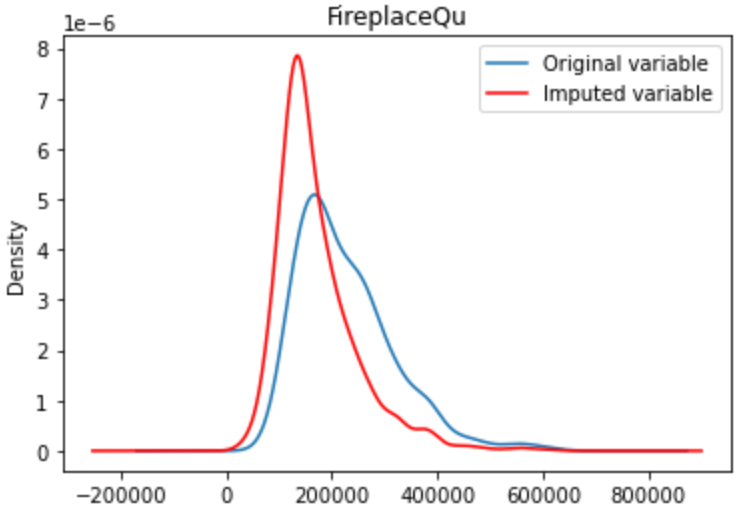
Out[17]:
```
<AxesSubplot:>
```



In [18]:
```
fig = plt.figure()
ax = fig.add_subplot(111)


temp.plot(kind='kde', ax=ax)

# distribution of the variable after imputation
df[df['FireplaceQu'] == 'Gd']['SalePrice'].plot(kind='kde', ax=ax, color='red')
```

```
lines, labels = ax.get_legend_handles_labels()
labels = ['Original variable', 'Imputed variable']
ax.legend(lines, labels, loc='best')

# add title
plt.title('FireplaceQu')
```

Out[18]:
```
Text(0.5, 1.0, 'FireplaceQu')
```



In [19]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(df.drop(columns=['SalePrice']),df['SalePr
```

In [20]:
```
from sklearn.impute import SimpleImputer
```

In [21]:
```
imputer = SimpleImputer(strategy='most_frequent')
```

In [22]:
```
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_train)
```

In [23]:
```
imputer.statistics_
```

Out[23]:
```
array(['Gd', 'TA'], dtype=object)
```

In [ ]:

Missing Indicator (automatically-select-imputer-parameters)

In [1]:
```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LogisticRegression
```

In [2]:
```
df = pd.read_csv('train.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	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 [4]:
```
df.drop(columns=['PassengerId','Name','Ticket','Cabin'],inplace=True)
```

In [5]:
```
#devide the columns
X = df.drop(columns=['Survived'])
y = df['Survived']
```

In [6]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [7]:
```
X_train.head()
```

Out[7]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
30	1	male	40.0	0	0	27.7208	C
10	3	female	4.0	1	1	16.7000	S
873	3	male	47.0	0	0	9.0000	S
182	3	male	9.0	4	2	31.3875	S
876	3	male	20.0	0	0	9.8458	S

In [8]:
```
y_train.head()
```

Out[8]:
```
30     0
10     1
873    0
182    0
876    0
Name: Survived, dtype: int64
```

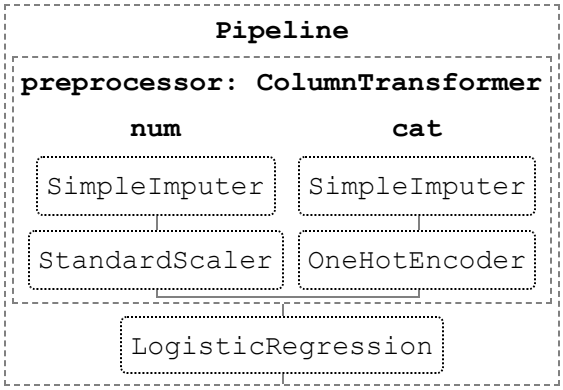
In [9]:
```
numerical_features = ['Age', 'Fare']
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_features = ['Embarked', 'Sex']
categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('ohe',OneHotEncoder(handle_unknown='ignore'))
])
```

In [10]:
```
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ]
)
```

In [11]:
```
clf = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression())
])
```

In [12]:
```
from sklearn import set_config

set_config(display='diagram')
clf
```

Out[12]:



In [13]:
```
param_grid = {
    'preprocessor__num__imputer__strategy': ['mean', 'median'],
    'preprocessor__cat__imputer__strategy': ['most_frequent', 'constant'],
    'classifier__C': [0.1, 1.0, 10, 100]
}

grid_search = GridSearchCV(clf, param_grid, cv=10)
```

In [14]:
```
grid_search.fit(X_train, y_train)

print(f"Best params:")
print(grid_search.best_params_)
```

```
Best params:
{'classifier__C': 1.0, 'preprocessor__cat__imputer__strategy': 'most_frequent', 'preproces
sor__num__imputer__strategy': 'mean'}
```

In [15]:
```
print(f"Internal CV score: {grid_search.best_score_:.3f}")
```

```
Internal CV score: 0.788
```

In [16]:
```
import pandas as pd

cv_results = pd.DataFrame(grid_search.cv_results_)
cv_results = cv_results.sort_values("mean_test_score", ascending=False)
cv_results[['param_classifier__C','param_preprocessor__cat__imputer__strategy','param_prep
```

Out[16]:

	param_classifier__C	param_preprocessor__cat__imputer__strategy	param_preprocessor__num__imputer__strategy	mea
4	1.0	most_frequent	mean	
5	1.0	most_frequent	median	
6	1.0	constant	mean	
7	1.0	constant	median	
8	10	most_frequent	mean	
9	10	most_frequent	median	
10	10	constant	mean	
11	10	constant	median	
12	100	most_frequent	mean	
13	100	most_frequent	median	
14	100	constant	mean	

	param_classifier__C	param_preprocessor__cat__imputer__strategy	param_preprocessor__num__imputer__strategy	mea
15	100	constant	median	
0	0.1	most_frequent	mean	
1	0.1	most_frequent	median	
2	0.1	constant	mean	
3	0.1	constant	median	

In [ ]:

Imputing Numerical Data (Man Median Imputation)

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
```

In [3]:
```
df = pd.read_csv('titanic_toy.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	Age	Fare	Family	Survived
0	22.0	7.2500	1	0
1	38.0	71.2833	1	1
2	26.0	7.9250	0	1
3	35.0	53.1000	1	1
4	35.0	8.0500	0	0

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Age       714 non-null    float64
 1   Fare      846 non-null    float64
 2   Family    891 non-null    int64  
 3   Survived  891 non-null    int64  
dtypes: float64(2), int64(2)
memory usage: 28.0 KB
```

In [6]:
```
df.isnull().mean()
```

Out[6]:
```
Age         0.198653
Fare        0.050505
Family      0.000000
Survived    0.000000
dtype: float64
```

In [7]:
```
X = df.drop(columns=['Survived'])
y = df['Survived']
```

In [8]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [9]:
```
X_train.shape, X_test.shape
```

Out[9]:
```
((712, 3), (179, 3))
```

In [10]:
```
X_train.isnull().mean()
```

Out[10]:
```
Age       0.207865
Fare      0.050562
Family    0.000000
dtype: float64
```

In [11]:
```
mean_age = X_train['Age'].mean()
median_age = X_train['Age'].median()

mean_fare = X_train['Fare'].mean()
median_fare = X_train['Fare'].median()
```

In [12]:
```
X_train['Age_median'] = X_train['Age'].fillna(median_age)
X_train['Age_mean'] = X_train['Age'].fillna(mean_age)

X_train['Fare_median'] = X_train['Fare'].fillna(median_fare)
X_train['Fare_mean'] = X_train['Fare'].fillna(mean_fare)
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_69128/2444989457.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train['Age_median'] = X_train['Age'].fillna(median_age)
C:\Users\HP\AppData\Local\Temp/ipykernel_69128/2444989457.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train['Age_mean'] = X_train['Age'].fillna(mean_age)
C:\Users\HP\AppData\Local\Temp/ipykernel_69128/2444989457.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train['Fare_median'] = X_train['Fare'].fillna(median_fare)
C:\Users\HP\AppData\Local\Temp/ipykernel_69128/2444989457.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  X_train['Fare_mean'] = X_train['Fare'].fillna(mean_fare)
```

In [13]:
```
X_train.sample(5)
```

Out[13]:

	Age	Fare	Family	Age_median	Age_mean	Fare_median	Fare_mean
154	NaN	7.3125	0	28.75	29.785904	7.3125	7.3125
580	25.0	30.0000	2	25.00	25.000000	30.0000	30.0000
747	30.0	13.0000	0	30.00	30.000000	13.0000	13.0000

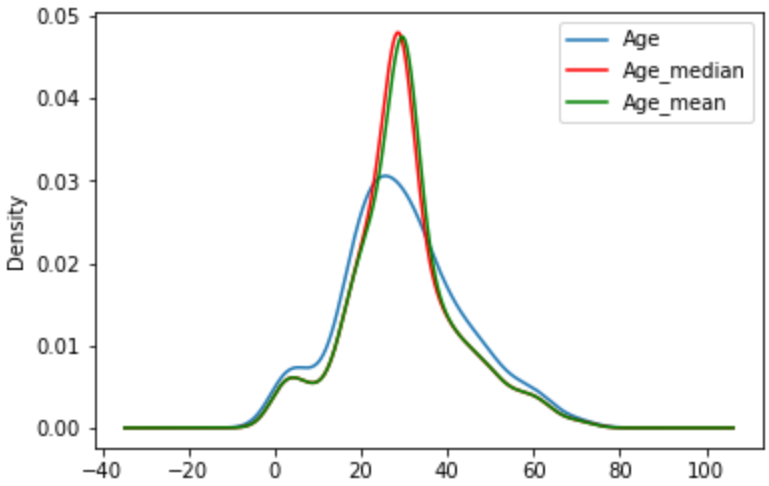
	Age	Fare	Family	Age_median	Age_mean	Fare_median	Fare_mean
292	36.0	12.8750	0	36.00	36.000000	12.8750	12.8750
435	14.0	120.0000	3	14.00	14.000000	120.0000	120.0000

In [14]:
```
print('Original Age variable variance: ', X_train['Age'].var())
print('Age Variance after median imputation: ', X_train['Age_median'].var())
print('Age Variance after mean imputation: ', X_train['Age_mean'].var())

print('Original Fare variable variance: ', X_train['Fare'].var())
print('Fare Variance after median imputation: ', X_train['Fare_median'].var())
print('Fare Variance after mean imputation: ', X_train['Fare_mean'].var())
```

```
Original Age variable variance:  204.3495133904614
Age Variance after median imputation:  161.9895663346054
Age Variance after mean imputation:  161.81262452718673
Original Fare variable variance:  2448.197913706318
Fare Variance after median imputation:  2340.0910219753637
Fare Variance after mean imputation:  2324.2385256705547
```

In [15]:
```
fig = plt.figure()
ax = fig.add_subplot(111)

# original variable distribution
X_train['Age'].plot(kind='kde', ax=ax)

# variable imputed with the median
X_train['Age_median'].plot(kind='kde', ax=ax, color='red')

# variable imputed with the mean
X_train['Age_mean'].plot(kind='kde', ax=ax, color='green')

# add legends
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

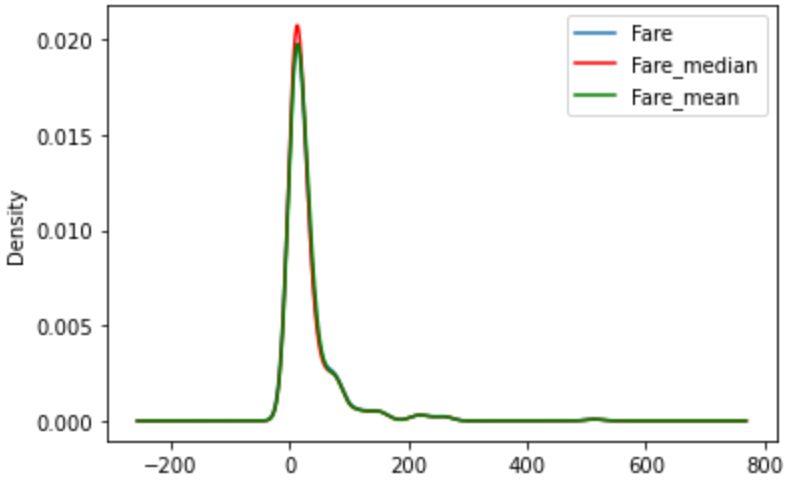
Out[15]:
```
<matplotlib.legend.Legend at 0x281a01a5850>
```



In [16]:
```
fig = plt.figure()
ax = fig.add_subplot(111)

# original variable distribution
X_train['Fare'].plot(kind='kde', ax=ax)

# variable imputed with the median
```

```
X_train['Fare_median'].plot(kind='kde', ax=ax, color='red')

# variable imputed with the mean
X_train['Fare_mean'].plot(kind='kde', ax=ax, color='green')

# add legends
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

Out[16]:
```
<matplotlib.legend.Legend at 0x281a09eff10>
```



In [17]:
```
X_train.cov()
```

Out[17]:

	Age	Fare	Family	Age_median	Age_mean	Fare_median	Fare_mean
Age	204.349513	70.719262	-6.498901	204.349513	204.349513	64.858859	66.665205
Fare	70.719262	2448.197914	17.258917	57.957599	55.603719	2448.197914	2448.197914
Family	-6.498901	17.258917	2.735252	-5.112563	-5.146106	16.476305	16.385048
Age_median	204.349513	57.957599	-5.112563	161.989566	161.812625	53.553455	55.023037
Age_mean	204.349513	55.603719	-5.146106	161.812625	161.812625	51.358000	52.788341
Fare_median	64.858859	2448.197914	16.476305	53.553455	51.358000	2340.091022	2324.238526
Fare_mean	66.665205	2448.197914	16.385048	55.023037	52.788341	2324.238526	2324.238526

In [18]:
```
X_train.corr()
```

Out[18]:

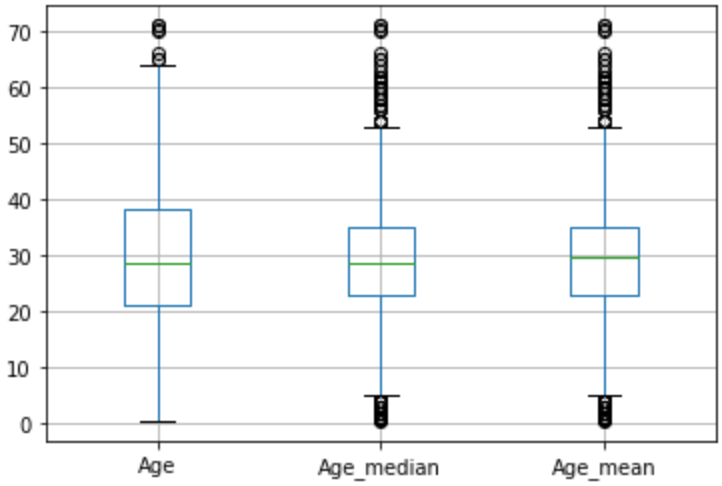
	Age	Fare	Family	Age_median	Age_mean	Fare_median	Fare_mean
Age	1.000000	0.092644	-0.299113	1.000000	1.000000	0.087356	0.090156
Fare	0.092644	1.000000	0.208268	0.091757	0.088069	1.000000	1.000000
Family	-0.299113	0.208268	1.000000	-0.242883	-0.244610	0.205942	0.205499
Age_median	1.000000	0.091757	-0.242883	1.000000	0.999454	0.086982	0.089673
Age_mean	1.000000	0.088069	-0.244610	0.999454	1.000000	0.083461	0.086078
Fare_median	0.087356	1.000000	0.205942	0.086982	0.083461	1.000000	0.996607
Fare_mean	0.090156	1.000000	0.205499	0.089673	0.086078	0.996607	1.000000

In [19]:

```
X_train[['Age', 'Age_median', 'Age_mean']].boxplot()
```

Out[19]:

```
<AxesSubplot:>
```

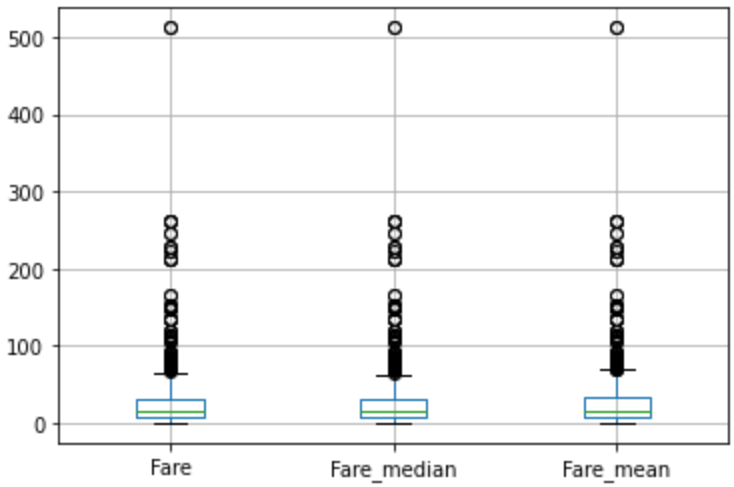


In [20]:

```
X_train[['Fare', 'Fare_median', 'Fare_mean']].boxplot()
```

Out[20]:

```
<AxesSubplot:>
```



Using Sklearn

In [21]:

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [22]:

```
imputer1 = SimpleImputer(strategy='median')
imputer2 = SimpleImputer(strategy='mean')
```

In [23]:

```
trf = ColumnTransformer([
    ('imputer1',imputer1,['Age']),
    ('imputer2',imputer2,['Fare'])
],remainder='passthrough')
```

In [24]:

```
trf.fit(X_train)
```

Out[24]:

```
ColumnTransformer(remainder='passthrough',
                  transformers=[('imputer1', SimpleImputer(strategy='median'),
                                 ['Age']),
                                ('imputer2', SimpleImputer(), ['Fare'])])
```

In [25]:
```
trf.named_transformers_['imputer1'].statistics_
```

Out[25]:
```
array([28.75])
```

In [26]:
```
trf.named_transformers_['imputer2'].statistics_
```

Out[26]:
```
array([32.61759689])
```

In [27]:
```
X_train = trf.transform(X_train)
X_test = trf.transform(X_test)
```

In [28]:
```
X_train
```

Out[28]:
```
array([[ 40.    ,  27.7208,   0.    ],
       [  4.    ,  16.7   ,   2.    ],
       [ 47.    ,   9.    ,   0.    ],
       ...,
       [ 71.    ,  49.5042,   0.    ],
       [ 28.75  , 221.7792,   0.    ],
       [ 28.75  ,  25.925 ,   0.    ]])
```

In [ ]:
```

```

Imputing Numerical Data (Arbitrary-Value-Imputation)

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
```

In [3]:
```
df = pd.read_csv('titanic_toy.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	Age	Fare	Family	Survived
0	22.0	7.2500	1	0
1	38.0	71.2833	1	1
2	26.0	7.9250	0	1
3	35.0	53.1000	1	1
4	35.0	8.0500	0	0

In [5]:
```
df.isnull().mean()
```

Out[5]:
```
Age         0.198653
Fare        0.050505
Family      0.000000
Survived    0.000000
dtype: float64
```

In [6]:
```
X = df.drop(columns=['Survived'])
y = df['Survived']
```

In [7]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [8]:
```
X_train['Age_99'] = X_train['Age'].fillna(99)
X_train['Age_minus1'] = X_train['Age'].fillna(-1)

X_train['Fare_999'] = X_train['Fare'].fillna(999)
X_train['Fare_minus1'] = X_train['Fare'].fillna(-1)
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_79272/3652012184.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  X_train['Age_99'] = X_train['Age'].fillna(99)
C:\Users\HP\AppData\Local\Temp/ipykernel_79272/3652012184.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  X_train['Age_minus1'] = X_train['Age'].fillna(-1)
C:\Users\HP\AppData\Local\Temp/ipykernel_79272/3652012184.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  X_train['Fare_999'] = X_train['Fare'].fillna(999)
C:\Users\HP\AppData\Local\Temp/ipykernel_79272/3652012184.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  X_train['Fare_minus1'] = X_train['Fare'].fillna(-1)
```

In [9]:

```
print('Original Age variable variance: ', X_train['Age'].var())
print('Age Variance after 99 wala imputation: ', X_train['Age_99'].var())
print('Age Variance after -1 wala imputation: ', X_train['Age_minus1'].var())

print('Original Fare variable variance: ', X_train['Fare'].var())
print('Fare Variance after 999 wala imputation: ', X_train['Fare_999'].var())
print('Fare Variance after -1 wala imputation: ', X_train['Fare_minus1'].var())
```

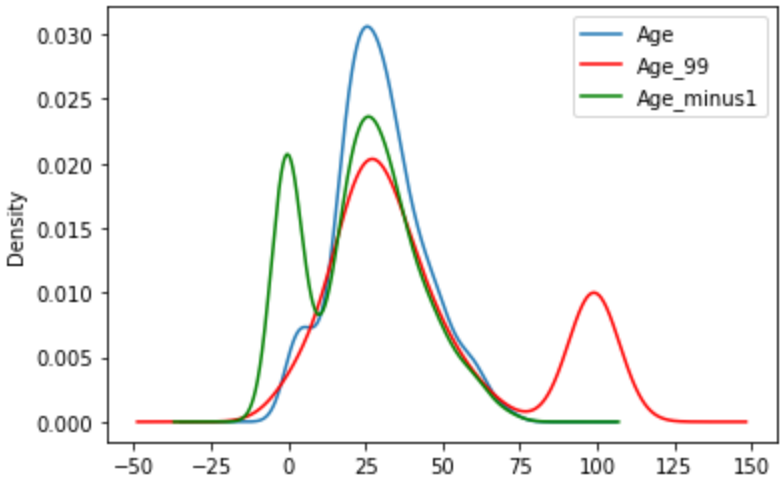
```
Original Age variable variance:  204.3495133904614
Age Variance after 99 wala imputation:  951.7275570187172
Age Variance after -1 wala imputation:  318.0896202624484
Original Fare variable variance:  2448.197913706318
Fare Variance after 999 wala imputation:  47219.20265217623
Fare Variance after -1 wala imputation:  2378.5676784883503
```

In [10]:

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original variable distribution
X_train['Age'].plot(kind='kde', ax=ax)

# variable imputed with the median
X_train['Age_99'].plot(kind='kde', ax=ax, color='red')

# variable imputed with the mean
X_train['Age_minus1'].plot(kind='kde', ax=ax, color='green')

# add legends
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

Out[10]:

```
<matplotlib.legend.Legend at 0x28930f0acd0>
```

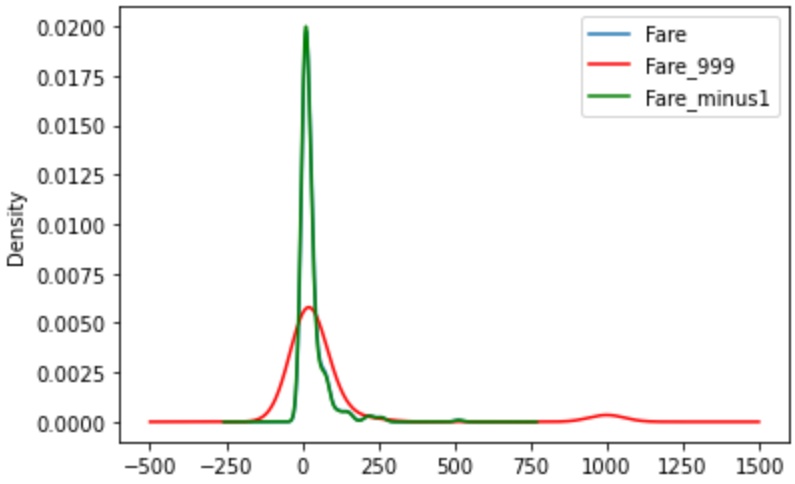


In [11]:

```
fig = plt.figure()
ax = fig.add_subplot(111)

# original variable distribution
X_train['Fare'].plot(kind='kde', ax=ax)

# variable imputed with the median
X_train['Fare_999'].plot(kind='kde', ax=ax, color='red')

# variable imputed with the mean
X_train['Fare_minus1'].plot(kind='kde', ax=ax, color='green')

# add legends
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

Out[11]:

```
<matplotlib.legend.Legend at 0x28931765760>
```



In [12]:

```
X_train.cov()
```

Out[12]:

	Age	Fare	Family	Age_99	Age_minus1	Fare_999	Fare_minus1
Age	204.349513	70.719262	-6.498901	204.349513	204.349513	162.793430	63.321188
Fare	70.719262	2448.197914	17.258917	-101.671097	125.558364	2448.197914	2448.197914
Family	-6.498901	17.258917	2.735252	-7.387287	-4.149246	11.528625	16.553989
Age_99	204.349513	-101.671097	-7.387287	951.727557	-189.535540	-159.931663	-94.317400
Age_minus1	204.349513	125.558364	-4.149246	-189.535540	318.089620	257.379887	114.394141

	Age	Fare	Family	Age_99	Age_minus1	Fare_999	Fare_minus1
Fare_999	162.793430	2448.197914	11.528625	-159.931663	257.379887	47219.202652	762.474982
Fare_minus1	63.321188	2448.197914	16.553989	-94.317400	114.394141	762.474982	2378.567678

In [13]:
```
X_train.corr()
```

Out[13]:

	Age	Fare	Family	Age_99	Age_minus1	Fare_999	Fare_minus1
Age	1.000000	0.092644	-0.299113	1.000000	1.000000	0.051179	0.084585
Fare	0.092644	1.000000	0.208268	-0.066273	0.142022	1.000000	1.000000
Family	-0.299113	0.208268	1.000000	-0.144787	-0.140668	0.032079	0.205233
Age_99	1.000000	-0.066273	-0.144787	1.000000	-0.344476	-0.023857	-0.062687
Age_minus1	1.000000	0.142022	-0.140668	-0.344476	1.000000	0.066411	0.131514
Fare_999	0.051179	1.000000	0.032079	-0.023857	0.066411	1.000000	0.071946
Fare_minus1	0.084585	1.000000	0.205233	-0.062687	0.131514	0.071946	1.000000

Using Sklearn

In [14]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [15]:
```
imputer1 = SimpleImputer(strategy='constant',fill_value=99)
imputer2 = SimpleImputer(strategy='constant',fill_value=999)
```

In [16]:
```
trf = ColumnTransformer([
    ('imputer1',imputer1,['Age']),
    ('imputer2',imputer2,['Fare'])
],remainder='passthrough')
```

In [17]:
```
trf.fit(X_train)
```

Out[17]:
```
ColumnTransformer(remainder='passthrough',
                  transformers=[('imputer1',
                                 SimpleImputer(fill_value=99,
                                               strategy='constant'),
                                 ['Age']),
                                ('imputer2',
                                 SimpleImputer(fill_value=999,
                                               strategy='constant'),
                                 ['Fare'])])
```

In [18]:
```
trf.named_transformers_['imputer1'].statistics_
```

Out[18]:
```
array([99.])
```

In [19]:
```
trf.named_transformers_['imputer2'].statistics_
```

Out[19]:
```
array([999.])
```

In [20]:
```
X_train = trf.transform(X_train)
X_test = trf.transform(X_test)
```

In [21]:
```
X_train
```

Out[21]:
```
array([[ 40.    ,  27.7208,   0.    ],
       [  4.    ,  16.7   ,   2.    ],
       [ 47.    ,   9.    ,   0.    ],
       ...,
       [ 71.    ,  49.5042,   0.    ],
       [ 99.    , 221.7792,   0.    ],
       [ 99.    ,  25.925 ,   0.    ]])
```

In [ ]:
```

```

Random Sample Imputation

In [1]:
```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('train.csv',usecols=['Age','Fare','Survived'])
```

In [3]:
```
df.head()
```

Out[3]:

	Survived	Age	Fare
0	0	22.0	7.2500
1	1	38.0	71.2833
2	1	26.0	7.9250
3	1	35.0	53.1000
4	0	35.0	8.0500

In [4]:
```
df.isnull().mean() * 100
```

Out[4]:
```
Survived     0.00000
Age         19.86532
Fare         0.00000
dtype: float64
```

In [5]:
```
X = df.drop(columns=['Survived'])
y = df['Survived']
```

In [6]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [7]:
```
X_train
```

Out[7]:

	Age	Fare
30	40.0	27.7208
10	4.0	16.7000
873	47.0	9.0000
182	9.0	31.3875
876	20.0	9.8458
...	...	...
534	30.0	8.6625

	Age	Fare
584	NaN	8.7125
493	71.0	49.5042
527	NaN	221.7792
168	NaN	25.9250

712 rows × 2 columns

In [8]:
```
X_train['Age_imputed'] = X_train['Age']
X_test['Age_imputed'] = X_test['Age']
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_80384/1230362693.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  X_train['Age_imputed'] = X_train['Age']
C:\Users\HP\AppData\Local\Temp/ipykernel_80384/1230362693.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  X_test['Age_imputed'] = X_test['Age']
```

In [9]:
```
X_test.tail()
```

Out[9]:

	Age	Fare	Age_imputed
89	24.0	8.0500	24.0
80	22.0	9.0000	22.0
846	NaN	69.5500	NaN
870	26.0	7.8958	26.0
251	29.0	10.4625	29.0

In [10]:
```
X_train['Age_imputed'][X_train['Age_imputed'].isnull()] = X_train['Age'].dropna().sample(X
X_test['Age_imputed'][X_test['Age_imputed'].isnull()] = X_train['Age'].dropna().sample(X_t
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py:8870: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  return self._update_inplace(result)
```

In [11]:
```
X_train['Age'].dropna().sample(X_train['Age'].isnull().sum()).values
```

Out[11]:
```
array([54.  , 50.  , 30.  , 59.  , 40.  , 22.  ,  4.  , 54.  , 12.  ,
       31.  , 45.  , 62.  , 40.5 , 20.  , 36.  , 48.  , 16.  , 44.  ,
       30.  , 36.  , 48.  , 14.  , 20.  , 24.  , 36.  , 27.  , 29.  ,
       14.  , 54.  , 23.  , 56.  , 50.  , 13.  , 11.  , 25.  , 30.  ,
```

```
 9.  , 16.  , 25.  , 45.  , 16.  , 65.  , 49.  , 70.5 , 50.  ,
42.  , 16.  , 19.  , 58.  , 26.  , 40.5 , 24.  , 34.  , 45.  ,
24.  , 24.  , 30.  , 32.  , 47.  , 14.5 , 29.  , 28.  , 30.  ,
17.  , 22.  , 62.  , 38.  , 22.  ,  0.42, 21.  , 33.  ,  2.  ,
32.5 , 18.  , 60.  , 21.  , 18.  ,  4.  , 28.5 , 70.  , 28.  ,
58.  , 34.  , 52.  , 44.  , 38.  ,  2.  , 21.  , 34.  , 18.  ,
54.  , 25.  , 33.  , 34.  , 62.  , 59.  ,  7.  , 28.  ,  0.75,
33.  , 20.  , 50.  , 32.  , 16.  , 23.5 , 36.  , 25.  , 28.  ,
32.  , 20.  , 17.  , 17.  , 25.  , 33.  , 27.  , 19.  , 51.  ,
39.  , 28.  , 32.  , 34.  , 22.  , 23.  , 35.  , 35.  , 22.  ,
23.  , 24.  , 35.  , 24.  , 45.  , 27.  , 27.  , 22.  , 42.  ,
21.  , 47.  , 45.5 , 41.  , 51.  , 37.  , 24.  , 39.  , 48.  ,
27.  , 30.  , 18.  ,  9.  ])
```

In [12]:
```
X_train['Age'].isnull().sum()
```

Out[12]:
```
148
```

In [13]:
```
X_train
```

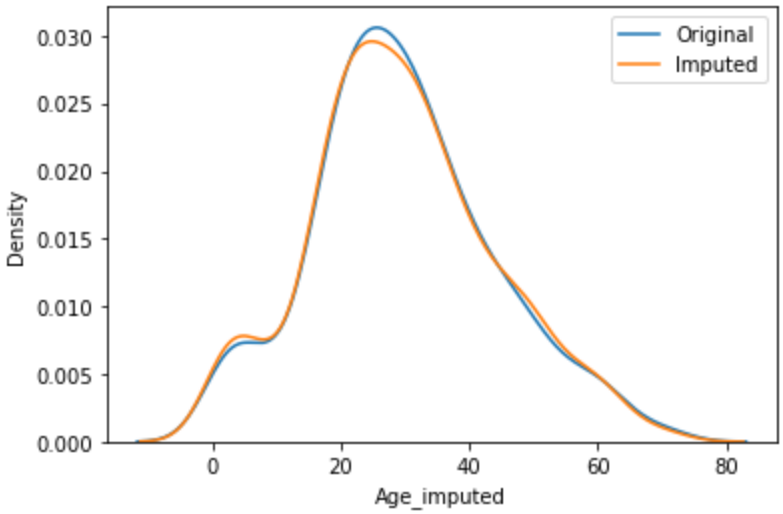
Out[13]:

	Age	Fare	Age_imputed
30	40.0	27.7208	40.0
10	4.0	16.7000	4.0
873	47.0	9.0000	47.0
182	9.0	31.3875	9.0
876	20.0	9.8458	20.0
...	...	...	...
534	30.0	8.6625	30.0
584	NaN	8.7125	59.0
493	71.0	49.5042	71.0
527	NaN	221.7792	34.0
168	NaN	25.9250	49.0

712 rows × 3 columns

In [14]:
```
sns.distplot(X_train['Age'],label='Original',hist=False)
sns.distplot(X_train['Age_imputed'],label = 'Imputed',hist=False)

plt.legend()
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `kd
eplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `kd
eplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



In [15]:
```
print('Original variable variance: ', X_train['Age'].var())
print('Variance after random imputation: ', X_train['Age_imputed'].var())
```

```
Original variable variance:  204.3495133904614
Variance after random imputation:  208.142956065796
```

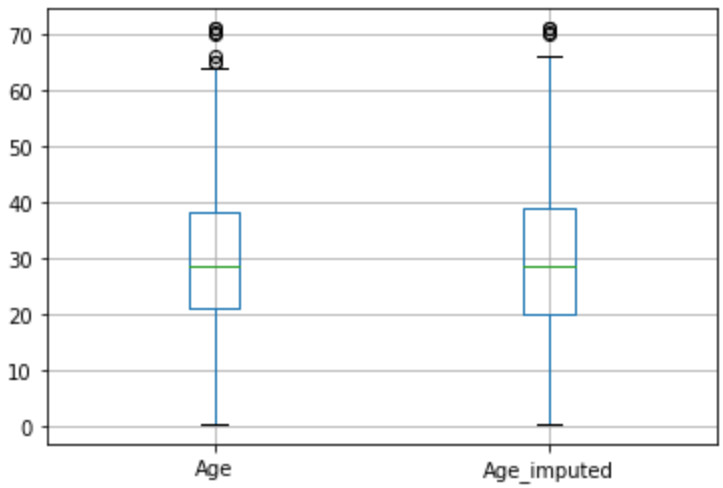
In [16]:
```
X_train[['Fare', 'Age', 'Age_imputed']].cov()
```

Out[16]:

	Fare	Age	Age_imputed
Fare	2368.246832	71.512440	58.923994
Age	71.512440	204.349513	204.349513
Age_imputed	58.923994	204.349513	208.142956

In [17]:
```
X_train[['Age', 'Age_imputed']].boxplot()
```

Out[17]:
```
<AxesSubplot:>
```



In [19]:
```
data = pd.read_csv('house-train.csv',usecols=['GarageQual','FireplaceQu', 'SalePrice'])
```

In [20]:
```
data.head()
```

Out[20]:

	FireplaceQu	GarageQual	SalePrice

	FireplaceQu	GarageQual	SalePrice
0	NaN	TA	208500
1	TA	TA	181500
2	TA	TA	223500
3	Gd	TA	140000
4	TA	TA	250000

In [21]:
```
data.isnull().mean() * 100
```

Out[21]:
```
FireplaceQu    47.260274
GarageQual      5.547945
SalePrice       0.000000
dtype: float64
```

In [22]:
```
X = data
y = data['SalePrice']
```

In [23]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [24]:
```
X_train['GarageQual_imputed'] = X_train['GarageQual']
X_test['GarageQual_imputed'] = X_test['GarageQual']

X_train['FireplaceQu_imputed'] = X_train['FireplaceQu']
X_test['FireplaceQu_imputed'] = X_test['FireplaceQu']
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_80384/3838090268.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  X_train['GarageQual_imputed'] = X_train['GarageQual']
C:\Users\HP\AppData\Local\Temp/ipykernel_80384/3838090268.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  X_test['GarageQual_imputed'] = X_test['GarageQual']
C:\Users\HP\AppData\Local\Temp/ipykernel_80384/3838090268.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  X_train['FireplaceQu_imputed'] = X_train['FireplaceQu']
C:\Users\HP\AppData\Local\Temp/ipykernel_80384/3838090268.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  X_test['FireplaceQu_imputed'] = X_test['FireplaceQu']
```

In [25]:
```
X_train.sample(5)
```

Out[25]:

	FireplaceQu	GarageQual	SalePrice	GarageQual_imputed	FireplaceQu_imputed
745	TA	TA	299800	TA	TA
300	Gd	TA	157000	TA	Gd
695	TA	TA	176000	TA	TA
1170	Po	TA	171000	TA	Po
1110	TA	TA	188000	TA	TA

In [26]:

```
X_train['GarageQual_imputed'][X_train['GarageQual_imputed'].isnull()] = X_train['GarageQua
X_test['GarageQual_imputed'][X_test['GarageQual_imputed'].isnull()] = X_train['GarageQual'

X_train['FireplaceQu_imputed'][X_train['FireplaceQu_imputed'].isnull()] = X_train['Firepla
X_test['FireplaceQu_imputed'][X_test['FireplaceQu_imputed'].isnull()] = X_train['Fireplace
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_80384/856878696.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  X_train['GarageQual_imputed'][X_train['GarageQual_imputed'].isnull()] = X_train['GarageQ
ual'].dropna().sample(X_train['GarageQual'].isnull().sum()).values
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py:8870: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  return self._update_inplace(result)
C:\Users\HP\AppData\Local\Temp/ipykernel_80384/856878696.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  X_test['GarageQual_imputed'][X_test['GarageQual_imputed'].isnull()] = X_train['GarageQua
l'].dropna().sample(X_test['GarageQual'].isnull().sum()).values
C:\Users\HP\AppData\Local\Temp/ipykernel_80384/856878696.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  X_train['FireplaceQu_imputed'][X_train['FireplaceQu_imputed'].isnull()] = X_train['Firep
laceQu'].dropna().sample(X_train['FireplaceQu'].isnull().sum()).values
C:\Users\HP\AppData\Local\Temp/ipykernel_80384/856878696.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_gu
ide/indexing.html#returning-a-view-versus-a-copy
  X_test['FireplaceQu_imputed'][X_test['FireplaceQu_imputed'].isnull()] = X_train['Firepla
ceQu'].dropna().sample(X_test['FireplaceQu'].isnull().sum()).values
```

In [27]:

```
temp = pd.concat(
        [
            X_train['GarageQual'].value_counts() / len(X_train['GarageQual'].dropna()),
            X_train['GarageQual_imputed'].value_counts() / len(X_train)
        ],
        axis=1)

temp.columns = ['original', 'imputed']
```

In [28]:

```
temp
```

Out[28]:

	original	imputed
TA	0.951043	0.952055
Fa	0.037171	0.035959
Gd	0.009973	0.010274
Po	0.000907	0.000856
Ex	0.000907	0.000856

In [29]:

```
temp = pd.concat(
        [
            X_train['FireplaceQu'].value_counts() / len(X_train['FireplaceQu'].dropna()),
            X_train['FireplaceQu_imputed'].value_counts() / len(df)
        ],
        axis=1)

temp.columns = ['original', 'imputed']

temp
```

Out[29]:

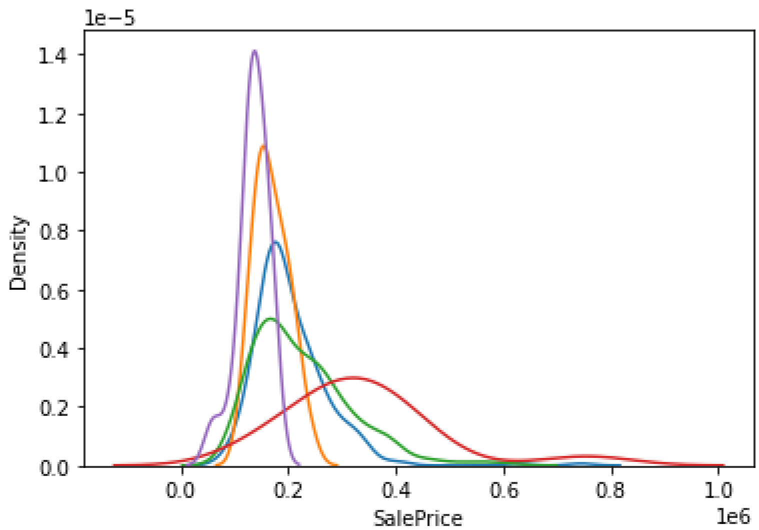
	original	imputed
Gd	0.494272	0.647587
TA	0.412439	0.543210
Fa	0.040917	0.052750
Po	0.027823	0.035915
Ex	0.024550	0.031425

In [30]:

```
for category in X_train['FireplaceQu'].dropna().unique():
    sns.distplot(X_train[X_train['FireplaceQu'] == category]['SalePrice'],hist=False,label
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `kd
eplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `kd
eplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `kd
eplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `kd
eplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
```

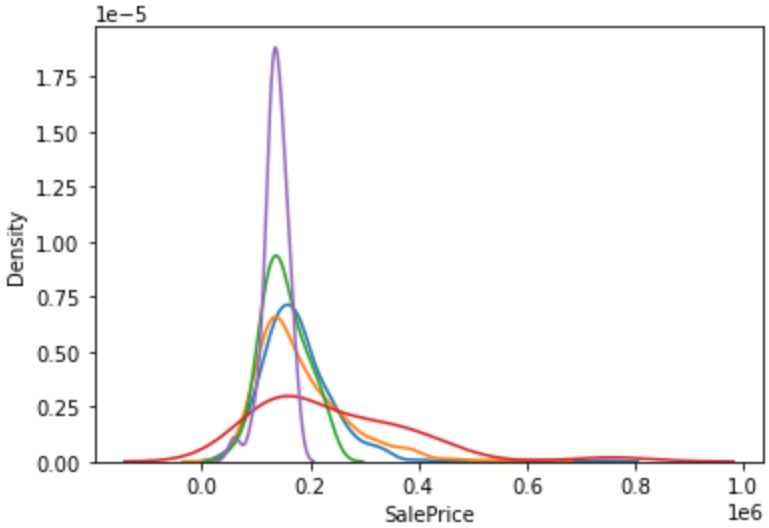
```
our code to use either `displot` (a figure-level function with similar flexibility) or `kd
eplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



In [31]:

```
for category in X_train['FireplaceQu_imputed'].dropna().unique():
    sns.distplot(X_train[X_train['FireplaceQu_imputed'] == category]['SalePrice'],hist=Fal
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `kd
eplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `kd
eplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `kd
eplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `kd
eplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. Please adapt y
our code to use either `displot` (a figure-level function with similar flexibility) or `kd
eplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```


1e−5
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Density
0.0
0.2
0.4
0.6
0.8
1.0
1e6
SalePrice

In [ ]:

KNN Imputer

In [1]:
```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.impute import KNNImputer,SimpleImputer
from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score
```

In [2]:
```
df = pd.read_csv('train.csv')[['Age','Pclass','Fare','Survived']]
```

In [3]:
```
df.head()
```

Out[3]:

	Age	Pclass	Fare	Survived
0	22.0	3	7.2500	0
1	38.0	1	71.2833	1
2	26.0	3	7.9250	1
3	35.0	1	53.1000	1
4	35.0	3	8.0500	0

In [4]:
```
df.isnull().mean() * 100
```

Out[4]:
```
Age         19.86532
Pclass       0.00000
Fare         0.00000
Survived     0.00000
dtype: float64
```

In [5]:
```
X = df.drop(columns=['Survived'])
y = df['Survived']
```

In [6]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
```

In [7]:
```
X_train.head()
```

Out[7]:

	Age	Pclass	Fare
30	40.0	1	27.7208
10	4.0	3	16.7000
873	47.0	3	9.0000
182	9.0	3	31.3875
876	20.0	3	9.8458

In [8]:
```
knn = KNNImputer(n_neighbors=3,weights='distance')

X_train_trf = knn.fit_transform(X_train)
X_test_trf = knn.transform(X_test)
```

In [9]:
```
lr = LogisticRegression()

lr.fit(X_train_trf,y_train)

y_pred = lr.predict(X_test_trf)

accuracy_score(y_test,y_pred)
```

Out[9]:
```
0.7150837988826816
```

In [10]:
```
# Comparision with Simple Imputer --> mean

si = SimpleImputer()

X_train_trf2 = si.fit_transform(X_train)
X_test_trf2 = si.transform(X_test)
```

In [11]:
```
lr = LogisticRegression()

lr.fit(X_train_trf2,y_train)

y_pred2 = lr.predict(X_test_trf2)

accuracy_score(y_test,y_pred2)
```

Out[11]:
```
0.6927374301675978
```

In [ ]:

In [ ]: