Copy_of_Lab_8_Data_Pre_processing_Classwork

November 4, 2024

Necessary Libraries

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

Loading the datasets

```
[ ]: !unzip Titanic\ Dataset.zip
```

```
Archive:  Titanic Dataset.zip
  inflating: Titanic Dataset/train.csv
  inflating: Titanic Dataset/test.csv
```

```
[ ]: !mv Titanic\ Dataset/* .
```

```
[ ]: test_data = pd.read_csv('test.csv')
train_data = pd.read_csv('train.csv')
```

```
[ ]: train_data.head()
```

```
[ ]:    PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th…  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

[ ]:
```
X = train_data.drop(columns=['Survived','Ticket', 'PassengerId',␣
↪'Cabin','Name'])
```

[ ]:
```
y = train_data['Survived']
```

Imputation (Fill in missing values)

[ ]:
```
from sklearn.impute import SimpleImputer
import numpy as np
```

[ ]:
```
X.isna().sum()
```

[ ]:
```
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

[ ]:
```
X.Age
```

[ ]:
```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       …
886    27.0
887    19.0
888     NaN
889    26.0
890    32.0
Name: Age, Length: 891, dtype: float64
```

[ ]:
```
imp_numerical = SimpleImputer(missing_values=np.nan, strategy='mean')
```

[ ]:
```
imp_numerical.fit(X[['Age']])
```

[ ]:
```
SimpleImputer()
```

[ ]:
```
X['Age'] = imp_numerical.transform(X[['Age']])
```

[ ]:
```
X['Age'].isna().sum()
```

[ ]:
```
0
```

[ ]:
```
X['Embarked'].unique()
```

[ ]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

[ ]:
```
imp_categorical = SimpleImputer(missing_values = np.nan,␣
↪strategy='most_frequent')
```

[ ]:
```
imp_categorical.fit(X[['Embarked']])
```

[ ]:
```
SimpleImputer(strategy='most_frequent')
```

[ ]:
```
X['Embarked'] = imp_categorical.transform(X[['Embarked']]).ravel()
```

[ ]:
```
X['Embarked'].isna().sum()
```

[ ]:
```
0
```

Standardization and Scaling

Three ways to do it: Standard Scaler, Minmax Scaler and Robust Scaler

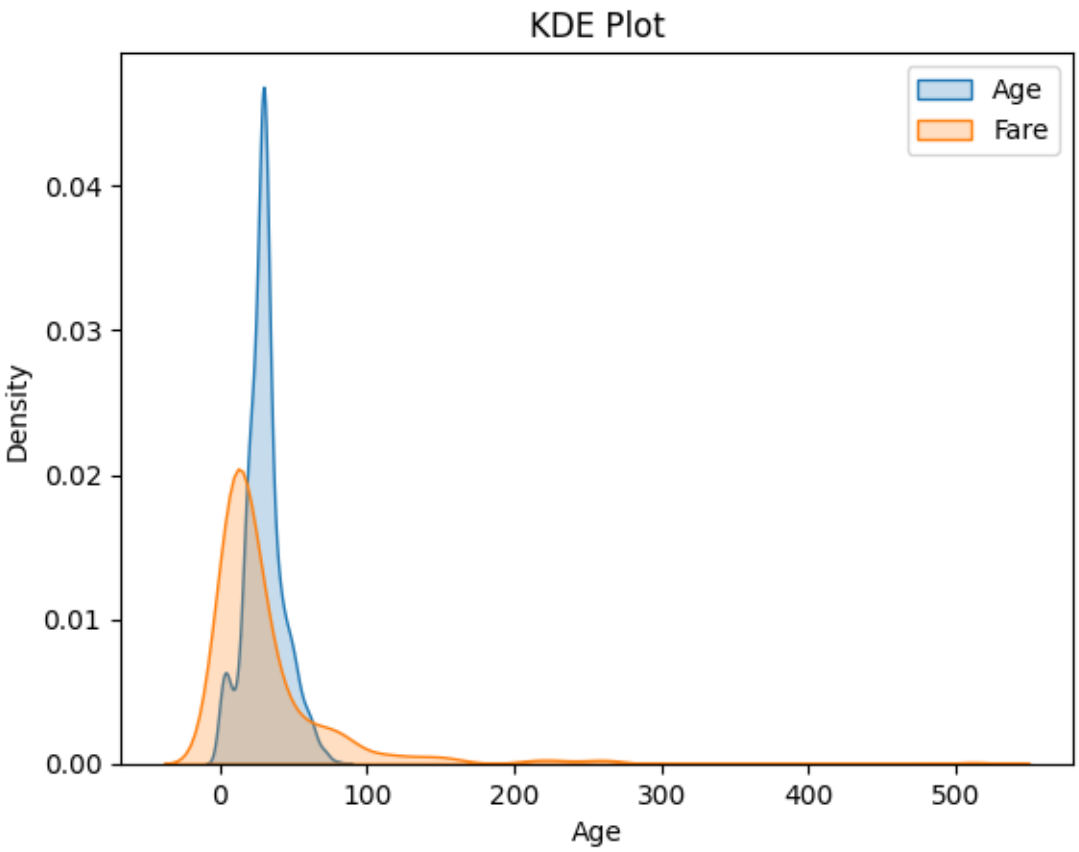
[ ]:
```
X.head()
```

[ ]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	male	22.0	1	0	7.2500	S
1	1	female	38.0	1	0	71.2833	C
2	3	female	26.0	0	0	7.9250	S
3	1	female	35.0	1	0	53.1000	S
4	3	male	35.0	0	0	8.0500	S

[ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
sns.kdeplot(X['Age'], fill=True, label='Age')
sns.kdeplot(X['Fare'], fill=True, label='Fare')

# Set labels and title
# plt.xlabel('Age')
plt.ylabel('Density')
plt.title('KDE Plot')
plt.legend()
# Display the KDE plot
plt.show()
```



[ ]:
```
import scipy.stats as stats
stats.probplot(X['Fare'], dist="norm", plot=plt)
```

[ ]:
```
((array([-3.16416595e+00, -2.89636677e+00, -2.74675222e+00, -2.64114608e+00,
         -2.55870259e+00, -2.49067391e+00, -2.43252738e+00, -2.38160005e+00,
         -2.33618969e+00, -2.29513992e+00, -2.25762808e+00, -2.22304736e+00,
         -2.19093694e+00, -2.16093830e+00, -2.13276686e+00, -2.10619283e+00,
         -2.08102787e+00, -2.05711563e+00, -2.03432484e+00, -2.01254418e+00,
         -1.99167841e+00, -1.97164537e+00, -1.95237369e+00, -1.93380097e+00,
         -1.91587229e+00, -1.89853909e+00, -1.88175821e+00, -1.86549107e+00,
         -1.84970311e+00, -1.83436318e+00, -1.81944313e+00, -1.80491744e+00,
         -1.79076290e+00, -1.77695830e+00, -1.76348425e+00, -1.75032296e+00,
         -1.73745806e+00, -1.72487445e+00, -1.71255819e+00, -1.70049636e+00,
         -1.68867698e+00, -1.67708890e+00, -1.66572174e+00, -1.65456583e+00,
         -1.64361212e+00, -1.63285214e+00, -1.62227795e+00, -1.61188210e+00,
         -1.60165759e+00, -1.59159783e+00, -1.58169661e+00, -1.57194806e+00,
         -1.56234666e+00, -1.55288715e+00, -1.54356460e+00, -1.53437430e+00,
         -1.52531180e+00, -1.51637287e+00, -1.50755349e+00, -1.49884983e+00,
         -1.49025825e+00, -1.48177530e+00, -1.47339765e+00, -1.46512215e+00,
         -1.45694579e+00, -1.44886569e+00, -1.44087909e+00, -1.43298336e+00,
```

-1.42517596e+00, -1.41745449e+00, -1.40981661e+00, -1.40226010e+00,
-1.39478282e+00, -1.38738272e+00, -1.38005783e+00, -1.37280624e+00,
-1.36562613e+00, -1.35851574e+00, -1.35147338e+00, -1.34449742e+00,
-1.33758628e+00, -1.33073844e+00, -1.32395244e+00, -1.31722687e+00,
-1.31056036e+00, -1.30395159e+00, -1.29739929e+00, -1.29090221e+00,
-1.28445918e+00, -1.27806903e+00, -1.27173065e+00, -1.26544295e+00,
-1.25920489e+00, -1.25301544e+00, -1.24687363e+00, -1.24077850e+00,
-1.23472912e+00, -1.22872458e+00, -1.22276403e+00, -1.21684660e+00,
-1.21097148e+00, -1.20513786e+00, -1.19934497e+00, -1.19359205e+00,
-1.18787837e+00, -1.18220320e+00, -1.17656585e+00, -1.17096565e+00,
-1.16540194e+00, -1.15987407e+00, -1.15438141e+00, -1.14892337e+00,
-1.14349933e+00, -1.13810873e+00, -1.13275101e+00, -1.12742560e+00,
-1.12213197e+00, -1.11686961e+00, -1.11163799e+00, -1.10643663e+00,
-1.10126502e+00, -1.09612271e+00, -1.09100921e+00, -1.08592409e+00,
-1.08086689e+00, -1.07583719e+00, -1.07083455e+00, -1.06585857e+00,
-1.06090885e+00, -1.05598498e+00, -1.05108658e+00, -1.04621327e+00,
-1.04136468e+00, -1.03654045e+00, -1.03174023e+00, -1.02696366e+00,
-1.02221041e+00, -1.01748014e+00, -1.01277253e+00, -1.00808726e+00,
-1.00342401e+00, -9.98782481e-01, -9.94162371e-01, -9.89563385e-01,
-9.84985234e-01, -9.80427636e-01, -9.75890312e-01, -9.71372991e-01,
-9.66875406e-01, -9.62397294e-01, -9.57938398e-01, -9.53498468e-01,
-9.49077254e-01, -9.44674515e-01, -9.40290012e-01, -9.35923511e-01,
-9.31574782e-01, -9.27243599e-01, -9.22929741e-01, -9.18632990e-01,
-9.14353133e-01, -9.10089959e-01, -9.05843261e-01, -9.01612837e-01,
-8.97398488e-01, -8.93200018e-01, -8.89017233e-01, -8.84849944e-01,
-8.80697966e-01, -8.76561115e-01, -8.72439211e-01, -8.68332076e-01,
-8.64239537e-01, -8.60161422e-01, -8.56097563e-01, -8.52047793e-01,
-8.48011949e-01, -8.43989871e-01, -8.39981400e-01, -8.35986380e-01,
-8.32004658e-01, -8.28036084e-01, -8.24080508e-01, -8.20137785e-01,
-8.16207769e-01, -8.12290320e-01, -8.08385297e-01, -8.04492562e-01,
-8.00611980e-01, -7.96743417e-01, -7.92886742e-01, -7.89041824e-01,
-7.85208535e-01, -7.81386750e-01, -7.77576344e-01, -7.73777194e-01,
-7.69989180e-01, -7.66212182e-01, -7.62446084e-01, -7.58690769e-01,
-7.54946122e-01, -7.51212033e-01, -7.47488388e-01, -7.43775079e-01,
-7.40071997e-01, -7.36379036e-01, -7.32696091e-01, -7.29023057e-01,
-7.25359833e-01, -7.21706316e-01, -7.18062408e-01, -7.14428009e-01,
-7.10803023e-01, -7.07187353e-01, -7.03580904e-01, -6.99983584e-01,
-6.96395299e-01, -6.92815959e-01, -6.89245472e-01, -6.85683751e-01,
-6.82130707e-01, -6.78586254e-01, -6.75050305e-01, -6.71522777e-01,
-6.68003585e-01, -6.64492646e-01, -6.60989880e-01, -6.57495204e-01,
-6.54008541e-01, -6.50529809e-01, -6.47058933e-01, -6.43595834e-01,
-6.40140437e-01, -6.36692665e-01, -6.33252446e-01, -6.29819705e-01,
-6.26394370e-01, -6.22976369e-01, -6.19565630e-01, -6.16162083e-01,
-6.12765660e-01, -6.09376290e-01, -6.05993906e-01, -6.02618441e-01,
-5.99249828e-01, -5.95888002e-01, -5.92532897e-01, -5.89184448e-01,
-5.85842593e-01, -5.82507267e-01, -5.79178409e-01, -5.75855957e-01,
-5.72539849e-01, -5.69230025e-01, -5.65926425e-01, -5.62628991e-01,

-5.59337662e-01, -5.56052382e-01, -5.52773092e-01, -5.49499736e-01,
-5.46232257e-01, -5.42970600e-01, -5.39714708e-01, -5.36464529e-01,
-5.33220006e-01, -5.29981087e-01, -5.26747718e-01, -5.23519846e-01,
-5.20297421e-01, -5.17080388e-01, -5.13868699e-01, -5.10662301e-01,
-5.07461145e-01, -5.04265181e-01, -5.01074359e-01, -4.97888630e-01,
-4.94707946e-01, -4.91532260e-01, -4.88361523e-01, -4.85195688e-01,
-4.82034708e-01, -4.78878538e-01, -4.75727130e-01, -4.72580440e-01,
-4.69438423e-01, -4.66301033e-01, -4.63168227e-01, -4.60039959e-01,
-4.56916187e-01, -4.53796868e-01, -4.50681957e-01, -4.47571414e-01,
-4.44465194e-01, -4.41363258e-01, -4.38265562e-01, -4.35172066e-01,
-4.32082729e-01, -4.28997511e-01, -4.25916370e-01, -4.22839267e-01,
-4.19766163e-01, -4.16697019e-01, -4.13631794e-01, -4.10570451e-01,
-4.07512950e-01, -4.04459255e-01, -4.01409326e-01, -3.98363127e-01,
-3.95320620e-01, -3.92281768e-01, -3.89246534e-01, -3.86214882e-01,
-3.83186776e-01, -3.80162179e-01, -3.77141055e-01, -3.74123371e-01,
-3.71109089e-01, -3.68098175e-01, -3.65090595e-01, -3.62086314e-01,
-3.59085297e-01, -3.56087510e-01, -3.53092920e-01, -3.50101494e-01,
-3.47113197e-01, -3.44127996e-01, -3.41145859e-01, -3.38166753e-01,
-3.35190645e-01, -3.32217503e-01, -3.29247294e-01, -3.26279988e-01,
-3.23315551e-01, -3.20353953e-01, -3.17395163e-01, -3.14439148e-01,
-3.11485878e-01, -3.08535323e-01, -3.05587451e-01, -3.02642232e-01,
-2.99699637e-01, -2.96759634e-01, -2.93822194e-01, -2.90887286e-01,
-2.87954883e-01, -2.85024953e-01, -2.82097468e-01, -2.79172399e-01,
-2.76249717e-01, -2.73329392e-01, -2.70411397e-01, -2.67495702e-01,
-2.64582279e-01, -2.61671101e-01, -2.58762138e-01, -2.55855364e-01,
-2.52950750e-01, -2.50048268e-01, -2.47147891e-01, -2.44249592e-01,
-2.41353343e-01, -2.38459117e-01, -2.35566887e-01, -2.32676627e-01,
-2.29788309e-01, -2.26901906e-01, -2.24017393e-01, -2.21134743e-01,
-2.18253928e-01, -2.15374924e-01, -2.12497705e-01, -2.09622243e-01,
-2.06748513e-01, -2.03876490e-01, -2.01006147e-01, -1.98137460e-01,
-1.95270402e-01, -1.92404948e-01, -1.89541074e-01, -1.86678753e-01,
-1.83817960e-01, -1.80958672e-01, -1.78100862e-01, -1.75244505e-01,
-1.72389578e-01, -1.69536055e-01, -1.66683912e-01, -1.63833125e-01,
-1.60983668e-01, -1.58135518e-01, -1.55288649e-01, -1.52443039e-01,
-1.49598663e-01, -1.46755497e-01, -1.43913516e-01, -1.41072697e-01,
-1.38233017e-01, -1.35394450e-01, -1.32556974e-01, -1.29720565e-01,
-1.26885199e-01, -1.24050853e-01, -1.21217504e-01, -1.18385126e-01,
-1.15553699e-01, -1.12723197e-01, -1.09893598e-01, -1.07064879e-01,
-1.04237016e-01, -1.01409987e-01, -9.85837679e-02, -9.57583359e-02,
-9.29336682e-02, -9.01097418e-02, -8.72865338e-02, -8.44640214e-02,
-8.16421817e-02, -7.88209919e-02, -7.60004293e-02, -7.31804713e-02,
-7.03610950e-02, -6.75422780e-02, -6.47239975e-02, -6.19062310e-02,
-5.90889559e-02, -5.62721498e-02, -5.34557901e-02, -5.06398543e-02,
-4.78243200e-02, -4.50091648e-02, -4.21943662e-02, -3.93799019e-02,
-3.65657495e-02, -3.37518867e-02, -3.09382910e-02, -2.81249403e-02,
-2.53118122e-02, -2.24988843e-02, -1.96861345e-02, -1.68735404e-02,
-1.40610798e-02, -1.12487304e-02, -8.43646994e-03, -5.62427623e-03,

```
-2.81212699e-03,  0.00000000e+00,  2.81212699e-03,  5.62427623e-03,
 8.43646994e-03,  1.12487304e-02,  1.40610798e-02,  1.68735404e-02,
 1.96861345e-02,  2.24988843e-02,  2.53118122e-02,  2.81249403e-02,
 3.09382910e-02,  3.37518867e-02,  3.65657495e-02,  3.93799019e-02,
 4.21943662e-02,  4.50091648e-02,  4.78243200e-02,  5.06398543e-02,
 5.34557901e-02,  5.62721498e-02,  5.90889559e-02,  6.19062310e-02,
 6.47239975e-02,  6.75422780e-02,  7.03610950e-02,  7.31804713e-02,
 7.60004293e-02,  7.88209919e-02,  8.16421817e-02,  8.44640214e-02,
 8.72865338e-02,  9.01097418e-02,  9.29336682e-02,  9.57583359e-02,
 9.85837679e-02,  1.01409987e-01,  1.04237016e-01,  1.07064879e-01,
 1.09893598e-01,  1.12723197e-01,  1.15553699e-01,  1.18385126e-01,
 1.21217504e-01,  1.24050853e-01,  1.26885199e-01,  1.29720565e-01,
 1.32556974e-01,  1.35394450e-01,  1.38233017e-01,  1.41072697e-01,
 1.43913516e-01,  1.46755497e-01,  1.49598663e-01,  1.52443039e-01,
 1.55288649e-01,  1.58135518e-01,  1.60983668e-01,  1.63833125e-01,
 1.66683912e-01,  1.69536055e-01,  1.72389578e-01,  1.75244505e-01,
 1.78100862e-01,  1.80958672e-01,  1.83817960e-01,  1.86678753e-01,
 1.89541074e-01,  1.92404948e-01,  1.95270402e-01,  1.98137460e-01,
 2.01006147e-01,  2.03876490e-01,  2.06748513e-01,  2.09622243e-01,
 2.12497705e-01,  2.15374924e-01,  2.18253928e-01,  2.21134743e-01,
 2.24017393e-01,  2.26901906e-01,  2.29788309e-01,  2.32676627e-01,
 2.35566887e-01,  2.38459117e-01,  2.41353343e-01,  2.44249592e-01,
 2.47147891e-01,  2.50048268e-01,  2.52950750e-01,  2.55855364e-01,
 2.58762138e-01,  2.61671101e-01,  2.64582279e-01,  2.67495702e-01,
 2.70411397e-01,  2.73329392e-01,  2.76249717e-01,  2.79172399e-01,
 2.82097468e-01,  2.85024953e-01,  2.87954883e-01,  2.90887286e-01,
 2.93822194e-01,  2.96759634e-01,  2.99699637e-01,  3.02642232e-01,
 3.05587451e-01,  3.08535323e-01,  3.11485878e-01,  3.14439148e-01,
 3.17395163e-01,  3.20353953e-01,  3.23315551e-01,  3.26279988e-01,
 3.29247294e-01,  3.32217503e-01,  3.35190645e-01,  3.38166753e-01,
 3.41145859e-01,  3.44127996e-01,  3.47113197e-01,  3.50101494e-01,
 3.53092920e-01,  3.56087510e-01,  3.59085297e-01,  3.62086314e-01,
 3.65090595e-01,  3.68098175e-01,  3.71109089e-01,  3.74123371e-01,
 3.77141055e-01,  3.80162179e-01,  3.83186776e-01,  3.86214882e-01,
 3.89246534e-01,  3.92281768e-01,  3.95320620e-01,  3.98363127e-01,
 4.01409326e-01,  4.04459255e-01,  4.07512950e-01,  4.10570451e-01,
 4.13631794e-01,  4.16697019e-01,  4.19766163e-01,  4.22839267e-01,
 4.25916370e-01,  4.28997511e-01,  4.32082729e-01,  4.35172066e-01,
 4.38265562e-01,  4.41363258e-01,  4.44465194e-01,  4.47571414e-01,
 4.50681957e-01,  4.53796868e-01,  4.56916187e-01,  4.60039959e-01,
 4.63168227e-01,  4.66301033e-01,  4.69438423e-01,  4.72580440e-01,
 4.75727130e-01,  4.78878538e-01,  4.82034708e-01,  4.85195688e-01,
 4.88361523e-01,  4.91532260e-01,  4.94707946e-01,  4.97888630e-01,
 5.01074359e-01,  5.04265181e-01,  5.07461145e-01,  5.10662301e-01,
 5.13868699e-01,  5.17080388e-01,  5.20297421e-01,  5.23519846e-01,
 5.26747718e-01,  5.29981087e-01,  5.33220006e-01,  5.36464529e-01,
 5.39714708e-01,  5.42970600e-01,  5.46232257e-01,  5.49499736e-01,
```

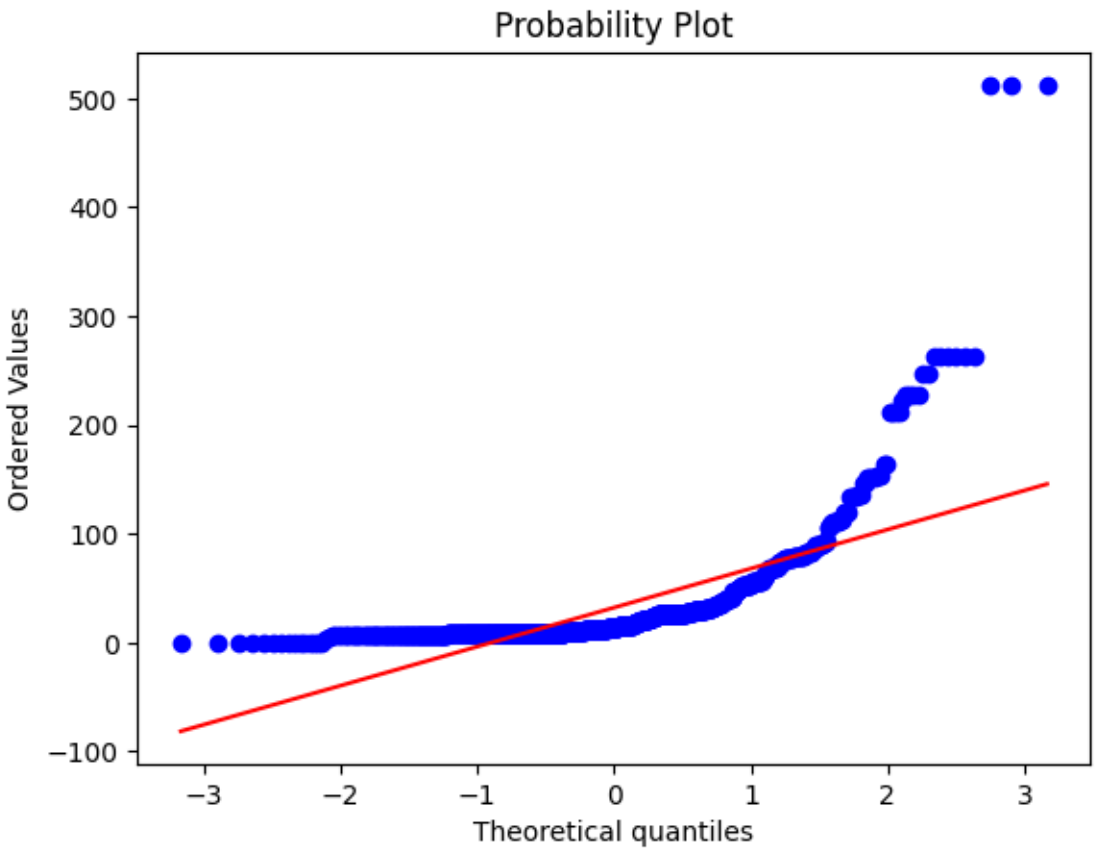
```
5.52773092e-01,  5.56052382e-01,  5.59337662e-01,  5.62628991e-01,
5.65926425e-01,  5.69230025e-01,  5.72539849e-01,  5.75855957e-01,
5.79178409e-01,  5.82507267e-01,  5.85842593e-01,  5.89184448e-01,
5.92532897e-01,  5.95888002e-01,  5.99249828e-01,  6.02618441e-01,
6.05993906e-01,  6.09376290e-01,  6.12765660e-01,  6.16162083e-01,
6.19565630e-01,  6.22976369e-01,  6.26394370e-01,  6.29819705e-01,
6.33252446e-01,  6.36692665e-01,  6.40140437e-01,  6.43595834e-01,
6.47058933e-01,  6.50529809e-01,  6.54008541e-01,  6.57495204e-01,
6.60989880e-01,  6.64492646e-01,  6.68003585e-01,  6.71522777e-01,
6.75050305e-01,  6.78586254e-01,  6.82130707e-01,  6.85683751e-01,
6.89245472e-01,  6.92815959e-01,  6.96395299e-01,  6.99983584e-01,
7.03580904e-01,  7.07187353e-01,  7.10803023e-01,  7.14428009e-01,
7.18062408e-01,  7.21706316e-01,  7.25359833e-01,  7.29023057e-01,
7.32696091e-01,  7.36379036e-01,  7.40071997e-01,  7.43775079e-01,
7.47488388e-01,  7.51212033e-01,  7.54946122e-01,  7.58690769e-01,
7.62446084e-01,  7.66212182e-01,  7.69989180e-01,  7.73777194e-01,
7.77576344e-01,  7.81386750e-01,  7.85208535e-01,  7.89041824e-01,
7.92886742e-01,  7.96743417e-01,  8.00611980e-01,  8.04492562e-01,
8.08385297e-01,  8.12290320e-01,  8.16207769e-01,  8.20137785e-01,
8.24080508e-01,  8.28036084e-01,  8.32004658e-01,  8.35986380e-01,
8.39981400e-01,  8.43989871e-01,  8.48011949e-01,  8.52047793e-01,
8.56097563e-01,  8.60161422e-01,  8.64239537e-01,  8.68332076e-01,
8.72439211e-01,  8.76561115e-01,  8.80697966e-01,  8.84849944e-01,
8.89017233e-01,  8.93200018e-01,  8.97398488e-01,  9.01612837e-01,
9.05843261e-01,  9.10089959e-01,  9.14353133e-01,  9.18632990e-01,
9.22929741e-01,  9.27243599e-01,  9.31574782e-01,  9.35923511e-01,
9.40290012e-01,  9.44674515e-01,  9.49077254e-01,  9.53498468e-01,
9.57938398e-01,  9.62397294e-01,  9.66875406e-01,  9.71372991e-01,
9.75890312e-01,  9.80427636e-01,  9.84985234e-01,  9.89563385e-01,
9.94162371e-01,  9.98782481e-01,  1.00342401e+00,  1.00808726e+00,
1.01277253e+00,  1.01748014e+00,  1.02221041e+00,  1.02696366e+00,
1.03174023e+00,  1.03654045e+00,  1.04136468e+00,  1.04621327e+00,
1.05108658e+00,  1.05598498e+00,  1.06090885e+00,  1.06585857e+00,
1.07083455e+00,  1.07583719e+00,  1.08086689e+00,  1.08592409e+00,
1.09100921e+00,  1.09612271e+00,  1.10126502e+00,  1.10643663e+00,
1.11163799e+00,  1.11686961e+00,  1.12213197e+00,  1.12742560e+00,
1.13275101e+00,  1.13810873e+00,  1.14349933e+00,  1.14892337e+00,
1.15438141e+00,  1.15987407e+00,  1.16540194e+00,  1.17096565e+00,
1.17656585e+00,  1.18220320e+00,  1.18787837e+00,  1.19359205e+00,
1.19934497e+00,  1.20513786e+00,  1.21097148e+00,  1.21684660e+00,
1.22276403e+00,  1.22872458e+00,  1.23472912e+00,  1.24077850e+00,
1.24687363e+00,  1.25301544e+00,  1.25920489e+00,  1.26544295e+00,
1.27173065e+00,  1.27806903e+00,  1.28445918e+00,  1.29090221e+00,
1.29739929e+00,  1.30395159e+00,  1.31056036e+00,  1.31722687e+00,
1.32395244e+00,  1.33073844e+00,  1.33758628e+00,  1.34449742e+00,
1.35147338e+00,  1.35851574e+00,  1.36562613e+00,  1.37280624e+00,
1.38005783e+00,  1.38738272e+00,  1.39478282e+00,  1.40226010e+00,
```

```
        1.40981661e+00,  1.41745449e+00,  1.42517596e+00,  1.43298336e+00,
        1.44087909e+00,  1.44886569e+00,  1.45694579e+00,  1.46512215e+00,
        1.47339765e+00,  1.48177530e+00,  1.49025825e+00,  1.49884983e+00,
        1.50755349e+00,  1.51637287e+00,  1.52531180e+00,  1.53437430e+00,
        1.54356460e+00,  1.55288715e+00,  1.56234666e+00,  1.57194806e+00,
        1.58169661e+00,  1.59159783e+00,  1.60165759e+00,  1.61188210e+00,
        1.62227795e+00,  1.63285214e+00,  1.64361212e+00,  1.65456583e+00,
        1.66572174e+00,  1.67708890e+00,  1.68867698e+00,  1.70049636e+00,
        1.71255819e+00,  1.72487445e+00,  1.73745806e+00,  1.75032296e+00,
        1.76348425e+00,  1.77695830e+00,  1.79076290e+00,  1.80491744e+00,
        1.81944313e+00,  1.83436318e+00,  1.84970311e+00,  1.86549107e+00,
        1.88175821e+00,  1.89853909e+00,  1.91587229e+00,  1.93380097e+00,
        1.95237369e+00,  1.97164537e+00,  1.99167841e+00,  2.01254418e+00,
        2.03432484e+00,  2.05711563e+00,  2.08102787e+00,  2.10619283e+00,
        2.13276686e+00,  2.16093830e+00,  2.19093694e+00,  2.22304736e+00,
        2.25762808e+00,  2.29513992e+00,  2.33618969e+00,  2.38160005e+00,
        2.43252738e+00,  2.49067391e+00,  2.55870259e+00,  2.64114608e+00,
        2.74675222e+00,  2.89636677e+00,  3.16416595e+00]),
array([ 0.    ,   0.    ,   0.    ,   0.    ,   0.    ,   0.    ,
        0.    ,   0.    ,   0.    ,   0.    ,   0.    ,   0.    ,
        0.    ,   0.    ,   0.    ,   4.0125,   5.    ,   6.2375,
        6.4375,   6.45  ,   6.4958,   6.4958,   6.75  ,   6.75  ,
        6.8583,   6.95  ,   6.975 ,   6.975 ,   7.0458,   7.05  ,
        7.05  ,   7.05  ,   7.05  ,   7.05  ,   7.05  ,   7.05  ,
        7.0542,   7.0542,   7.125 ,   7.125 ,   7.125 ,   7.125 ,
        7.1417,   7.225 ,   7.225 ,   7.225 ,   7.225 ,   7.225 ,
        7.225 ,   7.225 ,   7.225 ,   7.225 ,   7.225 ,   7.225 ,
        7.225 ,   7.2292,   7.2292,   7.2292,   7.2292,   7.2292,
        7.2292,   7.2292,   7.2292,   7.2292,   7.2292,   7.2292,
        7.2292,   7.2292,   7.2292,   7.2292,   7.25  ,   7.25  ,
        7.25  ,   7.25  ,   7.25  ,   7.25  ,   7.25  ,   7.25  ,
        7.25  ,   7.25  ,   7.25  ,   7.25  ,   7.25  ,   7.3125,
        7.4958,   7.4958,   7.4958,   7.5208,   7.55  ,   7.55  ,
        7.55  ,   7.55  ,   7.6292,   7.65  ,   7.65  ,   7.65  ,
        7.65  ,   7.725 ,   7.7292,   7.7333,   7.7333,   7.7333,
        7.7333,   7.7375,   7.7375,   7.7417,   7.75  ,   7.75  ,
        7.75  ,   7.75  ,   7.75  ,   7.75  ,   7.75  ,   7.75  ,
        7.75  ,   7.75  ,   7.75  ,   7.75  ,   7.75  ,   7.75  ,
        7.75  ,   7.75  ,   7.75  ,   7.75  ,   7.75  ,   7.75  ,
        7.75  ,   7.75  ,   7.75  ,   7.75  ,   7.75  ,   7.75  ,
        7.75  ,   7.75  ,   7.775 ,   7.775 ,   7.775 ,   7.775 ,
        7.775 ,   7.775 ,   7.775 ,   7.775 ,   7.775 ,   7.775 ,
        7.775 ,   7.775 ,   7.775 ,   7.775 ,   7.775 ,   7.775 ,
        7.7875,   7.7958,   7.7958,   7.7958,   7.7958,   7.7958,
        7.7958,   7.8   ,   7.8292,   7.8292,   7.8542,   7.8542,
        7.8542,   7.8542,   7.8542,   7.8542,   7.8542,   7.8542,
```

```
 7.8542,   7.8542,   7.8542,   7.8542,   7.8542,   7.875 ,
 7.8792,   7.8792,   7.8792,   7.8792,   7.8875,   7.8958,
 7.8958,   7.8958,   7.8958,   7.8958,   7.8958,   7.8958,
 7.8958,   7.8958,   7.8958,   7.8958,   7.8958,   7.8958,
 7.8958,   7.8958,   7.8958,   7.8958,   7.8958,   7.8958,
 7.8958,   7.8958,   7.8958,   7.8958,   7.8958,   7.8958,
 7.8958,   7.8958,   7.8958,   7.8958,   7.8958,   7.8958,
 7.8958,   7.8958,   7.8958,   7.8958,   7.8958,   7.8958,
 7.8958,   7.925 ,   7.925 ,   7.925 ,   7.925 ,   7.925 ,
 7.925 ,   7.925 ,   7.925 ,   7.925 ,   7.925 ,   7.925 ,
 7.925 ,   7.925 ,   7.925 ,   7.925 ,   7.925 ,   7.925 ,
 7.925 ,   8.0292,   8.05  ,   8.05  ,   8.05  ,   8.05  ,
 8.05  ,   8.05  ,   8.05  ,   8.05  ,   8.05  ,   8.05  ,
 8.05  ,   8.05  ,   8.05  ,   8.05  ,   8.05  ,   8.05  ,
 8.05  ,   8.05  ,   8.05  ,   8.05  ,   8.05  ,   8.05  ,
 8.05  ,   8.05  ,   8.05  ,   8.05  ,   8.05  ,   8.05  ,
 8.05  ,   8.05  ,   8.05  ,   8.05  ,   8.05  ,   8.05  ,
 8.05  ,   8.05  ,   8.05  ,   8.1125,   8.1375,   8.1583,
 8.3   ,   8.3625,   8.4042,   8.4333,   8.4583,   8.5167,
 8.6542,   8.6625,   8.6625,   8.6625,   8.6625,   8.6625,
 8.6625,   8.6625,   8.6625,   8.6625,   8.6625,   8.6625,
 8.6625,   8.6625,   8.6833,   8.7125,   8.85  ,   9.    ,
 9.    ,   9.2167,   9.225 ,   9.225 ,   9.35  ,   9.35  ,
 9.475 ,   9.4833,   9.5   ,   9.5   ,   9.5   ,   9.5   ,
 9.5   ,   9.5   ,   9.5   ,   9.5   ,   9.5   ,   9.5875,
 9.5875,   9.825 ,   9.825 ,   9.8375,   9.8417,   9.8458,
10.1708,  10.4625,  10.4625,  10.5   ,  10.5   ,  10.5   ,
10.5   ,  10.5   ,  10.5   ,  10.5   ,  10.5   ,  10.5   ,
10.5   ,  10.5   ,  10.5   ,  10.5   ,  10.5   ,  10.5   ,
10.5   ,  10.5   ,  10.5   ,  10.5   ,  10.5   ,  10.5   ,
10.5   ,  10.5   ,  10.5   ,  10.5167,  11.1333,  11.1333,
11.1333,  11.2417,  11.2417,  11.5   ,  11.5   ,  11.5   ,
11.5   ,  12.    ,  12.275 ,  12.2875,  12.35  ,  12.35  ,
12.35  ,  12.475 ,  12.475 ,  12.475 ,  12.475 ,  12.525 ,
12.65  ,  12.875 ,  13.    ,  13.    ,  13.    ,  13.    ,
13.    ,  13.    ,  13.    ,  13.    ,  13.    ,  13.    ,
13.    ,  13.    ,  13.    ,  13.    ,  13.    ,  13.    ,
13.    ,  13.    ,  13.    ,  13.    ,  13.    ,  13.    ,
13.    ,  13.    ,  13.    ,  13.    ,  13.    ,  13.    ,
13.    ,  13.    ,  13.    ,  13.    ,  13.    ,  13.    ,
13.    ,  13.    ,  13.4167,  13.5   ,  13.5   ,  13.5   ,
13.5   ,  13.7917,  13.8583,  13.8625,  14.    ,  14.1083,
14.4   ,  14.4   ,  14.4542,  14.4542,  14.4542,  14.4542,
14.4542,  14.4542,  14.4542,  14.4583,  14.4583,  14.4583,
14.5   ,  14.5   ,  14.5   ,  14.5   ,  14.5   ,  14.5   ,
```

```
14.5   ,  15.    ,  15.0458,  15.05  ,  15.1   ,  15.2458,
15.2458,  15.2458,  15.2458,  15.2458,  15.5   ,  15.5   ,
15.5   ,  15.5   ,  15.5   ,  15.5   ,  15.5   ,  15.5   ,
15.55  ,  15.7417,  15.7417,  15.75  ,  15.85  ,  15.85  ,
15.85  ,  15.85  ,  15.9   ,  15.9   ,  16.    ,  16.1   ,
16.1   ,  16.1   ,  16.1   ,  16.1   ,  16.1   ,  16.1   ,
16.1   ,  16.1   ,  16.7   ,  16.7   ,  17.4   ,  17.8   ,
17.8   ,  18.    ,  18.    ,  18.    ,  18.75  ,  18.75  ,
18.75  ,  18.7875,  18.7875,  19.2583,  19.2583,  19.2583,
19.2583,  19.5   ,  19.5   ,  19.9667,  19.9667,  20.2125,
20.2125,  20.25  ,  20.25  ,  20.525 ,  20.525 ,  20.525 ,
20.575 ,  20.575 ,  21.    ,  21.    ,  21.    ,  21.    ,
21.    ,  21.    ,  21.075 ,  21.075 ,  21.075 ,  21.075 ,
21.6792,  22.025 ,  22.3583,  22.3583,  22.525 ,  23.    ,
23.    ,  23.    ,  23.    ,  23.25  ,  23.25  ,  23.45  ,
23.45  ,  24.    ,  24.    ,  24.15  ,  24.15  ,  24.15  ,
24.15  ,  24.15  ,  24.15  ,  24.15  ,  24.15  ,  25.4667,
25.4667,  25.4667,  25.4667,  25.5875,  25.925 ,  25.9292,
25.9292,  26.    ,  26.    ,  26.    ,  26.    ,  26.    ,
26.    ,  26.    ,  26.    ,  26.    ,  26.    ,  26.    ,
26.    ,  26.    ,  26.    ,  26.    ,  26.    ,  26.    ,
26.    ,  26.    ,  26.    ,  26.    ,  26.    ,  26.    ,
26.    ,  26.    ,  26.    ,  26.    ,  26.    ,  26.    ,
26.    ,  26.    ,  26.25  ,  26.25  ,  26.25  ,  26.25  ,
26.25  ,  26.25  ,  26.2833,  26.2875,  26.2875,  26.2875,
26.3875,  26.55  ,  26.55  ,  26.55  ,  26.55  ,  26.55  ,
26.55  ,  26.55  ,  26.55  ,  26.55  ,  26.55  ,  26.55  ,
26.55  ,  26.55  ,  26.55  ,  26.55  ,  27.    ,  27.    ,
27.7208,  27.7208,  27.7208,  27.7208,  27.7208,  27.75  ,
27.75  ,  27.75  ,  27.75  ,  27.9   ,  27.9   ,  27.9   ,
27.9   ,  27.9   ,  27.9   ,  28.5   ,  28.7125,  29.    ,
29.    ,  29.125 ,  29.125 ,  29.125 ,  29.125 ,  29.125 ,
29.7   ,  29.7   ,  29.7   ,  30.    ,  30.    ,  30.    ,
30.    ,  30.    ,  30.    ,  30.0708,  30.0708,  30.5   ,
30.5   ,  30.5   ,  30.5   ,  30.5   ,  30.6958,  30.6958,
31.    ,  31.    ,  31.    ,  31.275 ,  31.275 ,  31.275 ,
31.275 ,  31.275 ,  31.275 ,  31.275 ,  31.3875,  31.3875,
31.3875,  31.3875,  32.3208,  32.5   ,  33.    ,  33.    ,
33.    ,  33.5   ,  34.0208,  34.375 ,  34.375 ,  34.375 ,
34.375 ,  34.6542,  35.    ,  35.5   ,  35.5   ,  35.5   ,
35.5   ,  36.75  ,  36.75  ,  37.0042,  37.0042,  38.5   ,
39.    ,  39.    ,  39.    ,  39.    ,  39.4   ,  39.6   ,
39.6   ,  39.6875,  39.6875,  39.6875,  39.6875,  39.6875,
39.6875,  40.125 ,  41.5792,  41.5792,  41.5792,  42.4   ,
46.9   ,  46.9   ,  46.9   ,  46.9   ,  46.9   ,  46.9   ,
47.1   ,  49.5   ,  49.5042,  49.5042,  50.    ,  50.4958,
51.4792,  51.8625,  51.8625,  52.    ,  52.    ,  52.    ,
```

```
        52.    ,  52.    ,  52.    ,  52.    ,  52.5542,  52.5542,
        52.5542,  53.1   ,  53.1   ,  53.1   ,  53.1   ,  53.1   ,
        55.    ,  55.    ,  55.4417,  55.9   ,  55.9   ,  56.4958,
        56.4958,  56.4958,  56.4958,  56.4958,  56.4958,  56.4958,
        56.9292,  56.9292,  57.    ,  57.    ,  57.9792,  57.9792,
        59.4   ,  61.175 ,  61.3792,  61.9792,  63.3583,  65.    ,
        65.    ,  66.6   ,  66.6   ,  69.3   ,  69.3   ,  69.55  ,
        69.55  ,  69.55  ,  69.55  ,  69.55  ,  69.55  ,  69.55  ,
        71.    ,  71.    ,  71.2833,  73.5   ,  73.5   ,  73.5   ,
        73.5   ,  73.5   ,  75.25  ,  76.2917,  76.7292,  76.7292,
        76.7292,  77.2875,  77.2875,  77.9583,  77.9583,  77.9583,
        78.2667,  78.2667,  78.85  ,  78.85  ,  79.2   ,  79.2   ,
        79.2   ,  79.2   ,  79.65  ,  79.65  ,  79.65  ,  80.    ,
        80.    ,  81.8583,  82.1708,  82.1708,  83.1583,  83.1583,
        83.1583,  83.475 ,  83.475 ,  86.5   ,  86.5   ,  86.5   ,
        89.1042,  89.1042,  90.    ,  90.    ,  90.    ,  90.    ,
        91.0792,  91.0792,  93.5   ,  93.5   , 106.425 , 106.425 ,
       108.9   , 108.9   , 110.8833, 110.8833, 110.8833, 110.8833,
       113.275 , 113.275 , 113.275 , 120.    , 120.    , 120.    ,
       120.    , 133.65  , 133.65  , 134.5   , 134.5   , 135.6333,
       135.6333, 135.6333, 146.5208, 146.5208, 151.55  , 151.55  ,
       151.55  , 151.55  , 153.4625, 153.4625, 153.4625, 164.8667,
       164.8667, 211.3375, 211.3375, 211.3375, 211.5   , 221.7792,
       227.525 , 227.525 , 227.525 , 227.525 , 247.5208, 247.5208,
       262.375 , 262.375 , 263.    , 263.    , 263.    , 263.    ,
       512.3292, 512.3292, 512.3292])),
(35.891199361614504, 32.204207968574636, 0.7204363989046636))
```



[ ]:
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

[ ]:
```
standardScaler = StandardScaler()
```

[ ]:
```
standardScaler.fit(X[['Age']])
```

[ ]:
```
StandardScaler()
```

[ ]:
```
X['Age'] = standardScaler.transform(X[['Age']])
```

[ ]:
```
X['Age']
```

[ ]:
```
0     -0.592481
1      0.638789
2     -0.284663
3      0.407926
4      0.407926
         ...
886   -0.207709
887   -0.823344
```

```
888    0.000000
889   -0.284663
890    0.177063
Name: Age, Length: 891, dtype: float64
```

[ ]:
```
minmaxScaler = MinMaxScaler()
```

[ ]:
```
minmaxScaler.fit(X[['Fare']])
X['Fare'] = minmaxScaler.transform(X[['Fare']])
```

[ ]:
```
X['Fare']
```

[ ]:
```
0      0.014151
1      0.139136
2      0.015469
3      0.103644
4      0.015713
         …
886    0.025374
887    0.058556
888    0.045771
889    0.058556
890    0.015127
Name: Fare, Length: 891, dtype: float64
```

Encoding the categorical values

Two ways: Ordinal and OneHotEncoder

[ ]:
```
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
```

[ ]:
```
ordinal = OrdinalEncoder()
```

[ ]:
```
X['Sex'] = ordinal.fit_transform(X[['Sex']])
```

[ ]:
```
onehot = OneHotEncoder(sparse_output=True)
```

[ ]:
```
embarked = onehot.fit_transform(X[['Embarked']])
```

[ ]:
```
feature_names = onehot.get_feature_names_out()
```

[ ]:
```
feature_names
```

[ ]:
```
array(['Embarked_C', 'Embarked_Q', 'Embarked_S'], dtype=object)
```

[ ]:
```
embarked_df = pd.DataFrame(embarked.toarray(), columns=feature_names)
```

[ ]:
```
embarked_df
```

[ ]:
```
     Embarked_C  Embarked_Q  Embarked_S
0           0.0         0.0         1.0
1           1.0         0.0         0.0
2           0.0         0.0         1.0
3           0.0         0.0         1.0
4           0.0         0.0         1.0
..          …           …           …
886         0.0         0.0         1.0
887         0.0         0.0         1.0
888         0.0         0.0         1.0
889         1.0         0.0         0.0
890         0.0         1.0         0.0

[891 rows x 3 columns]
```

[ ]:
```
X = pd.concat([X, embarked_df],axis=1).drop(columns=['Embarked'])
```

Combining and Pipelining

[ ]:
```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
```

[ ]:
```
X_pipeline = train_data.drop(columns=['Survived','Ticket', 'PassengerId',␣
↪'Cabin','Name'])
```

[ ]:
```
age_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

fare_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', MinMaxScaler())
])

embarked_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder())
])

sex_transformer = Pipeline([
    ('encoder', OrdinalEncoder())
])

preprocessor = ColumnTransformer(
    transformers=[
        ('sex_transformer', sex_transformer, ['Sex']),
        ('age_transformer', age_transformer, ['Age']),
```

```
        ('fare_transformer', fare_transformer, ['Fare']),
        ('embarked_transformer', embarked_transformer, ['Embarked'])
    ],remainder='passthrough',)

pipeline = Pipeline([
    ('preprocessor', preprocessor),
])

X_transformed = pipeline.fit_transform(X_pipeline)
```

[ ]:
```
new_order = ['Sex','Age','Fare','Embarked_C',␣
 ↪'Embarked_Q',        'Embarked_S','Pclass', 'SibSp',        'Parch']
```

[ ]:
```
pd.DataFrame(X_transformed,columns=new_order)
```

[ ]:
```
      Sex       Age      Fare  Embarked_C  Embarked_Q  Embarked_S  Pclass  \
0     1.0 -0.592481  0.014151         0.0         0.0         1.0     3.0
1     0.0  0.638789  0.139136         1.0         0.0         0.0     1.0
2     0.0 -0.284663  0.015469         0.0         0.0         1.0     3.0
3     0.0  0.407926  0.103644         0.0         0.0         1.0     1.0
4     1.0  0.407926  0.015713         0.0         0.0         1.0     3.0
..    …       …       …          …           …           …      …
886   1.0 -0.207709  0.025374         0.0         0.0         1.0     2.0
887   0.0 -0.823344  0.058556         0.0         0.0         1.0     1.0
888   0.0  0.000000  0.045771         0.0         0.0         1.0     3.0
889   1.0 -0.284663  0.058556         1.0         0.0         0.0     1.0
890   1.0  0.177063  0.015127         0.0         1.0         0.0     3.0

     SibSp  Parch
0      1.0    0.0
1      1.0    0.0
2      0.0    0.0
3      1.0    0.0
4      0.0    0.0
..     …    …
886    0.0    0.0
887    0.0    0.0
888    1.0    2.0
889    0.0    0.0
890    0.0    0.0

[891 rows x 9 columns]
```

[ ]:
```
X.head(1)
```

[ ]:
```
   Pclass  Sex       Age  SibSp  Parch      Fare  Embarked_C  Embarked_Q  \
0       3  1.0 -0.592481      1      0  0.014151         0.0         0.0
```

```
   Embarked_S
0         1.0
```

0.1 Do the same for the Dataset in link: https://www.kaggle.com/datasets/sujithmandala/s loan-classification-dataset - [10 marks]

[ ]:
```
from google.colab import files
files.upload()
```

```
<IPython.core.display.HTML object>

Saving kaggle.json to kaggle.json
```

[ ]:
```
{'kaggle.json':
b'{"username":"ulugbekshernazarov","key":"04a6f70b9e9b10a3c9cfb00e5ddb14c2"}'}
```

[ ]:
```
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download sujithmandala/simple-loan-classification-dataset
```

```
Dataset URL: https://www.kaggle.com/datasets/sujithmandala/simple-loan-
classification-dataset
License(s): CC-BY-SA-4.0
Downloading simple-loan-classification-dataset.zip to /content
  0% 0.00/1.05k [00:00<?, ?B/s]
100% 1.05k/1.05k [00:00<00:00, 2.34MB/s]
```

[ ]:
```
!unzip simple-loan-classification-dataset.zip
```

```
Archive:  simple-loan-classification-dataset.zip
  inflating: loan.csv
```

[ ]:
```
df = pd.read_csv('loan.csv')
df.head()
```

[ ]:
```
   age  gender  occupation education_level marital_status  income  \
0   32    Male    Engineer      Bachelor's        Married   85000
1   45  Female     Teacher        Master's         Single   62000
2   28    Male     Student     High School         Single   25000
3   51  Female     Manager      Bachelor's        Married  105000
4   36    Male  Accountant      Bachelor's        Married   75000

   credit_score loan_status
0           720    Approved
1           680    Approved
2           590      Denied
3           780    Approved
```

```
4           710    Approved
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61 entries, 0 to 60
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   age              61 non-null     int64
 1   gender           61 non-null     object
 2   occupation       61 non-null     object
 3   education_level  61 non-null     object
 4   marital_status   61 non-null     object
 5   income           61 non-null     int64
 6   credit_score     61 non-null     int64
 7   loan_status      61 non-null     object
dtypes: int64(3), object(5)
memory usage: 3.9+ KB
```

[ ]:
```
df.describe()
```

[ ]:

	age	income	credit_score
count	61.000000	61.000000	61.000000
mean	37.081967	78983.606557	709.836066
std	8.424755	33772.025802	72.674888
min	24.000000	25000.000000	560.000000
25%	30.000000	52000.000000	650.000000
50%	36.000000	78000.000000	720.000000
75%	43.000000	98000.000000	770.000000
max	55.000000	180000.000000	830.000000

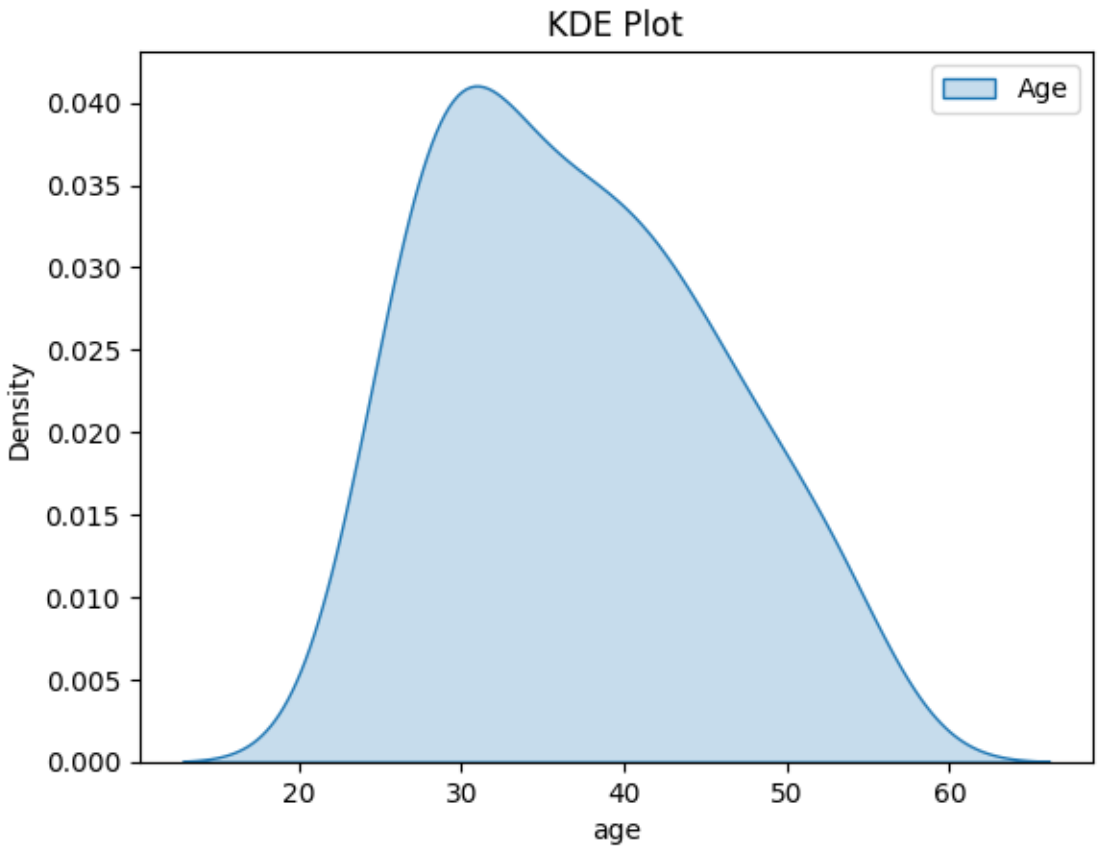
[ ]:
```
df.isna().sum()
```

[ ]:
```
age                0
gender             0
occupation         0
education_level    0
marital_status     0
income             0
credit_score       0
loan_status        0
dtype: int64
```

Standardization and Scaling

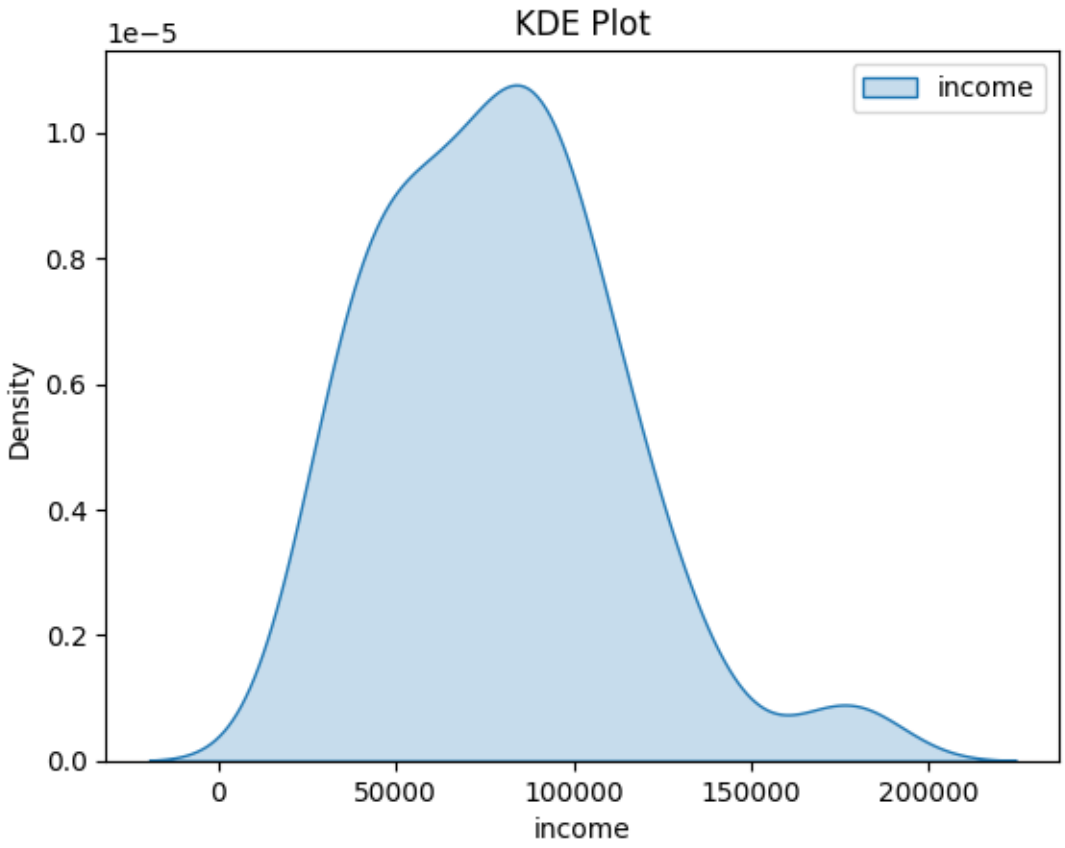
[ ]:
```
sns.kdeplot(df['age'], fill=True, label='Age')
# sns.kdeplot(df['income'], fill=True, label='income')
```

```
# sns.kdeplot(df['credit_score'], fill=True, label='credit_score')

# Set labels and title
# plt.xlabel('Age')
plt.ylabel('Density')
plt.title('KDE Plot')
plt.legend()
# Display the KDE plot
plt.show()
```

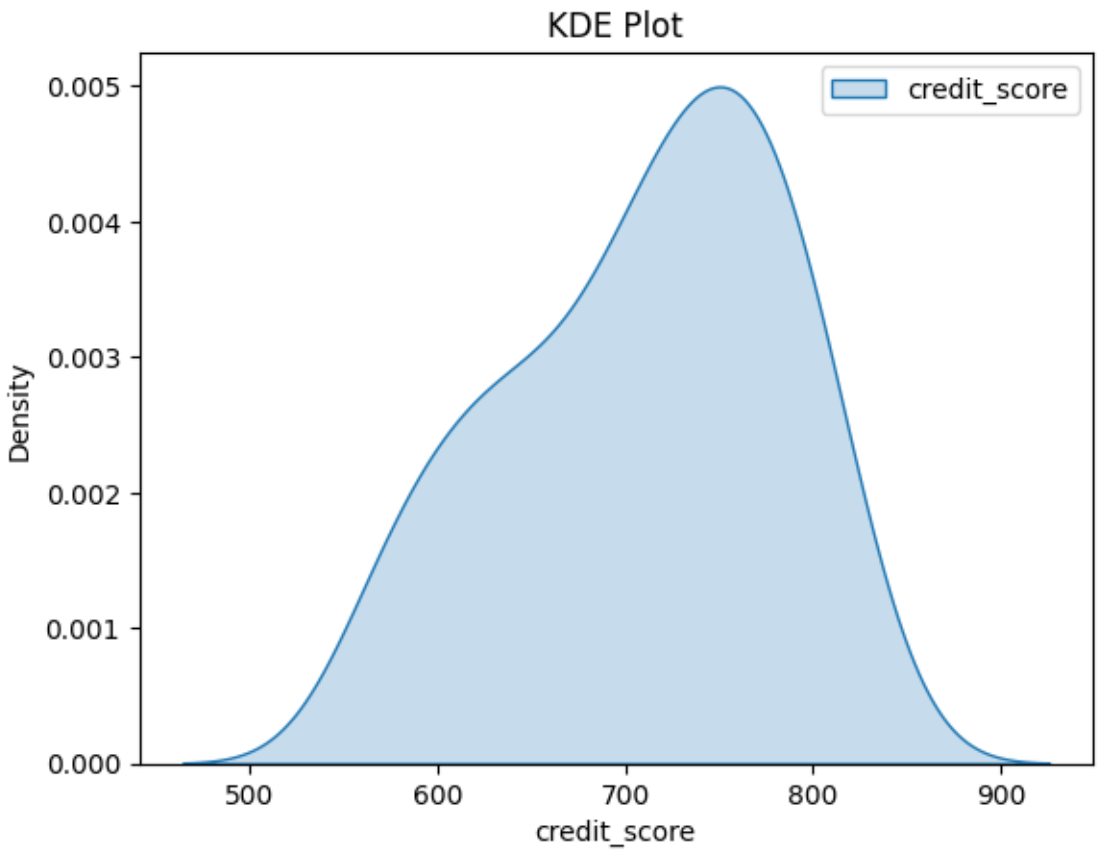


[ ]:
```
# sns.kdeplot(df['age'], fill=True, label='Age')
sns.kdeplot(df['income'], fill=True, label='income')
# sns.kdeplot(df['credit_score'], fill=True, label='credit_score')

# Set labels and title
# plt.xlabel('Age')
plt.ylabel('Density')
plt.title('KDE Plot')
plt.legend()
```

```
# Display the KDE plot
plt.show()
```



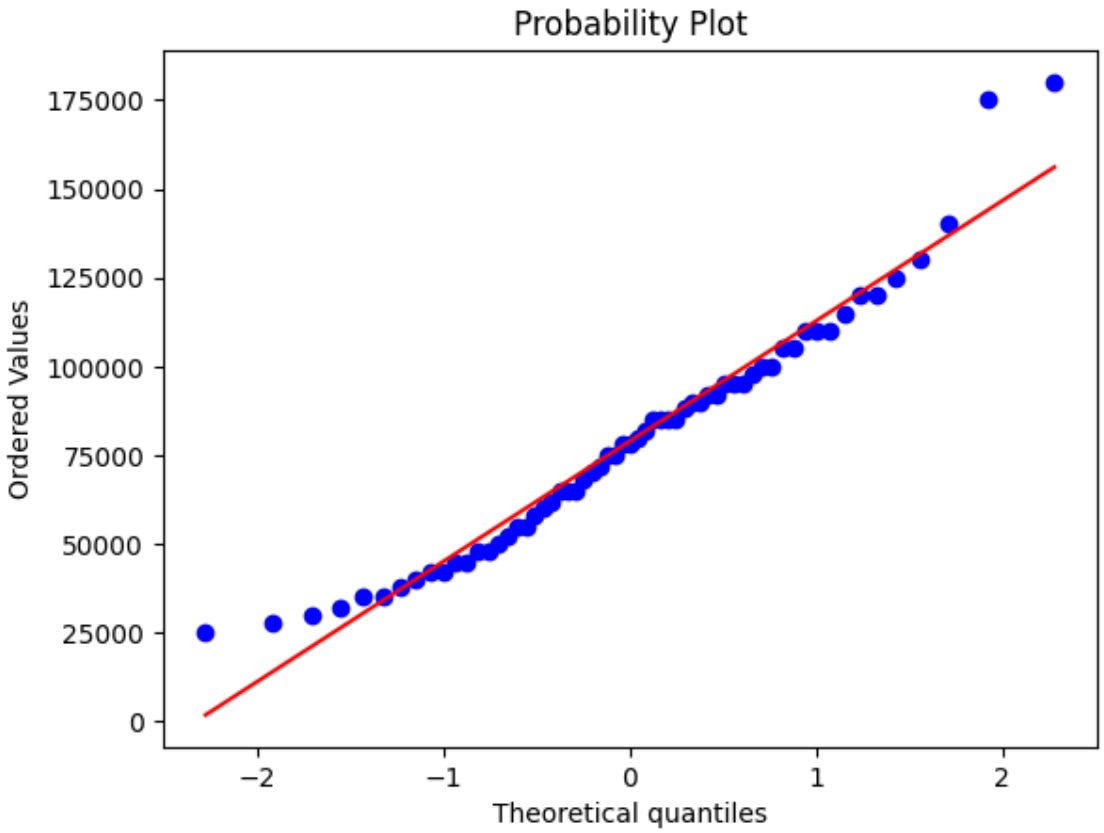
[ ]:
```
# sns.kdeplot(df['age'], fill=True, label='Age')
# sns.kdeplot(df['income'], fill=True, label='income')
sns.kdeplot(df['credit_score'], fill=True, label='credit_score')

# Set labels and title
# plt.xlabel('Age')
plt.ylabel('Density')
plt.title('KDE Plot')
plt.legend()
# Display the KDE plot
plt.show()
```



[ ]:
```
import scipy.stats as stats
stats.probplot(df['income'], dist="norm", plot=plt)
```

[ ]:
```
((array([-2.28017173, -1.92017484, -1.7091256 , -1.55469152, -1.43036801,
         -1.3249031 , -1.23241216, -1.14940983, -1.07365152, -1.00360143,
         -0.93815914, -0.87650695, -0.81801894, -0.76220403, -0.70866868,
         -0.65709167, -0.60720655, -0.55878903, -0.51164772, -0.46561731,
         -0.42055328, -0.37632784, -0.33282677, -0.28994682, -0.24759369,
         -0.20568029, -0.16412527, -0.12285188, -0.08178679, -0.04085922,
          0.        ,  0.04085922,  0.08178679,  0.12285188,  0.16412527,
          0.20568029,  0.24759369,  0.28994682,  0.33282677,  0.37632784,
          0.42055328,  0.46561731,  0.51164772,  0.55878903,  0.60720655,
          0.65709167,  0.70866868,  0.76220403,  0.81801894,  0.87650695,
          0.93815914,  1.00360143,  1.07365152,  1.14940983,  1.23241216,
          1.3249031 ,  1.43036801,  1.55469152,  1.7091256 ,  1.92017484,
          2.28017173]),
  array([ 25000,  28000,  30000,  32000,  35000,  35000,  38000,  40000,
          42000,  42000,  45000,  45000,  48000,  48000,  50000,  52000,
          55000,  55000,  58000,  60000,  62000,  65000,  65000,  65000,
          68000,  70000,  72000,  75000,  75000,  78000,  78000,  80000,
```

```
        82000,  85000,  85000,  85000,  85000,  88000,  90000,  90000,
        92000,  92000,  95000,  95000,  95000,  98000, 100000, 100000,
       105000, 105000, 110000, 110000, 110000, 115000, 120000, 120000,
       125000, 130000, 140000, 175000, 180000])),
(33819.21078234064, 78983.60655737705, 0.9792044418708735))
```



[ ]:
```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

[ ]:
```
standardScaler = StandardScaler()
```

[ ]:
```
standardScaler.fit(df[['age']])
```

[ ]:
```
StandardScaler()
```

[ ]:
```
df['age'] = standardScaler.transform(df[['age']])
```

[ ]:
```
df['age']
```

[ ]:
```
0    -0.608224
1     0.947653
```

```
2    -1.086956
3     1.665750
4    -0.129493
        …
56    0.229556
57   -1.446005
58    0.708287
59   -0.847590
60    0.109873
Name: age, Length: 61, dtype: float64
```

[ ]:
```
standardScaler.fit(df[['credit_score']])
df['credit_score'] = standardScaler.transform(df[['credit_score']])
df['credit_score']
```

[ ]:
```
0     0.141015
1    -0.413949
2    -1.662618
3     0.973462
4     0.002274
        …
56    0.834721
57   -1.940100
58    0.695980
59   -0.830172
60   -0.136467
Name: credit_score, Length: 61, dtype: float64
```

[ ]:
```
minmaxScaler = MinMaxScaler()
```

[ ]:
```
minmaxScaler.fit(df[['income']])
df['income'] = minmaxScaler.transform(df[['income']])
```

[ ]:
```
X['Fare']
```

[ ]:
```
0      0.014151
1      0.139136
2      0.015469
3      0.103644
4      0.015713
         …
886    0.025374
887    0.058556
888    0.045771
889    0.058556
890    0.015127
Name: Fare, Length: 891, dtype: float64
```

Encoding the categorical values

Two ways: Ordinal and OneHotEncoder

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

[ ]:
```
        age  gender  occupation education_level marital_status    income  \
0 -0.608224    Male    Engineer      Bachelor's        Married  0.387097
1  0.947653  Female     Teacher        Master's         Single  0.238710
2 -1.086956    Male     Student     High School         Single  0.000000
3  1.665750  Female     Manager      Bachelor's        Married  0.516129
4 -0.129493    Male  Accountant      Bachelor's        Married  0.322581

   credit_score loan_status
0      0.141015    Approved
1     -0.413949    Approved
2     -1.662618      Denied
3      0.973462    Approved
4      0.002274    Approved
```

[ ]:
```
from sklearn.preprocessing import OrdinalEncoder, OneHotEncoder
```

[ ]:
```
ordinal = OrdinalEncoder()
```

[ ]:
```
df['occupation'] = ordinal.fit_transform(df[['occupation']])
```

[ ]:
```
df['gender'] = ordinal.fit_transform(df[['gender']])
df['marital_status'] = ordinal.fit_transform(df[['marital_status']])
df['loan_status'] = ordinal.fit_transform(df[['loan_status']])
```

[ ]:
```
onehot = OneHotEncoder(sparse_output=True)
```

[ ]:
```
embarked = onehot.fit_transform(df[['education_level']])
```

[ ]:
```
feature_names = onehot.get_feature_names_out()
```

[ ]:
```
feature_names
```

[ ]:
```
array(["education_level_Associate's", "education_level_Bachelor's",
       'education_level_Doctoral', 'education_level_High School',
       "education_level_Master's"], dtype=object)
```

[ ]:
```
embarked_df = pd.DataFrame(embarked.toarray(), columns=feature_names)
```

[ ]:
```
embarked_df
```

[ ]:
```
    education_level_Associate's  education_level_Bachelor's  \
0                           0.0                         1.0
1                           0.0                         0.0
```

```
2                           0.0                           0.0
3                           0.0                           1.0
4                           0.0                           1.0
..                          …                             …
56                          0.0                           0.0
57                          0.0                           0.0
58                          0.0                           1.0
59                          0.0                           0.0
60                          1.0                           0.0

    education_level_Doctoral  education_level_High School  \
0                        0.0                           0.0
1                        0.0                           0.0
2                        0.0                           1.0
3                        0.0                           0.0
4                        0.0                           0.0
..                       …                             …
56                       0.0                           0.0
57                       0.0                           1.0
58                       0.0                           0.0
59                       0.0                           0.0
60                       0.0                           0.0

    education_level_Master's
0                        0.0
1                        1.0
2                        0.0
3                        0.0
4                        0.0
..                       …
56                       1.0
57                       0.0
58                       0.0
59                       1.0
60                       0.0

[61 rows x 5 columns]
```

[ ]:
```
df = pd.concat([df, embarked_df],axis=1).drop(columns=['education_level'])
```

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

[ ]:
```
        age  gender  occupation  marital_status    income  credit_score  \
0 -0.608224     1.0        12.0             0.0  0.387097      0.141015
1  0.947653     0.0        35.0             1.0  0.238710     -0.413949
2 -1.086956     1.0        33.0             1.0  0.000000     -1.662618
3  1.665750     0.0        16.0             0.0  0.516129      0.973462
```

```
4 -0.129493     1.0         0.0             0.0  0.322581      0.002274

   loan_status  education_level_Associate's  education_level_Bachelor's  \
0          0.0                          0.0                         1.0
1          0.0                          0.0                         0.0
2          1.0                          0.0                         0.0
3          0.0                          0.0                         1.0
4          0.0                          0.0                         1.0

   education_level_Doctoral  education_level_High School  \
0                       0.0                          0.0
1                       0.0                          0.0
2                       0.0                          1.0
3                       0.0                          0.0
4                       0.0                          0.0

   education_level_Master's
0                       0.0
1                       1.0
2                       0.0
3                       0.0
4                       0.0
```

Combining and Pipelining

[ ]:
```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
```

[ ]:
```
df = pd.read_csv('loan.csv')
df.head()
```

[ ]:
```
   age  gender  occupation education_level marital_status  income  \
0   32    Male    Engineer      Bachelor's        Married   85000
1   45  Female     Teacher        Master's         Single   62000
2   28    Male     Student     High School         Single   25000
3   51  Female     Manager      Bachelor's        Married  105000
4   36    Male  Accountant      Bachelor's        Married   75000

   credit_score loan_status
0           720    Approved
1           680    Approved
2           590      Denied
3           780    Approved
4           710    Approved
```

[ ]:
```
X_pipeline = df
```

[ ]:
```
age_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler())
])

fare_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', MinMaxScaler())
])

embarked_transformer = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('encoder', OneHotEncoder())
])

sex_transformer = Pipeline([
    ('encoder', OrdinalEncoder())
])

preprocessor = ColumnTransformer(
    transformers=[
        ('gender_transformer', sex_transformer, ['gender']),
        ('occupation_transformer', sex_transformer, ['occupation']),
        ('marital_status_transformer', sex_transformer, ['marital_status']),
        ('loan_status_status_transformer', sex_transformer, ['loan_status']),
        ('age_transformer', age_transformer, ['age']),
        ('income_transformer', fare_transformer, ['income']),
        ('credit_score_transformer', fare_transformer, ['credit_score']),
        ('education_level_status_transformer', embarked_transformer,␣
 ↪['education_level'])
    ],remainder='passthrough',)

pipeline = Pipeline([
    ('preprocessor', preprocessor),
])

X_transformed = pipeline.fit_transform(X_pipeline)
```

[ ]:
```
new_order = ['gender','occupation','marital_status', 'loan_status', 'age',␣
 ↪'income',        'credit_score','education_level_A',␣
 ↪'education_level_B',        'education_level_C', 'education_level_M',␣
 ↪'education_level_D']
```

[ ]:
```
df_train = pd.DataFrame(X_transformed,columns=new_order)
df_train.head()
```

[ ]:
```
   gender  occupation  marital_status  loan_status       age    income  \
0     1.0        12.0             0.0          0.0 -0.608224  0.387097
1     0.0        35.0             1.0          0.0  0.947653  0.238710
2     1.0        33.0             1.0          1.0 -1.086956  0.000000
3     0.0        16.0             0.0          0.0  1.665750  0.516129
4     1.0         0.0             0.0          0.0 -0.129493  0.322581

   credit_score  education_level_A  education_level_B  education_level_C  \
0      0.592593                0.0                1.0                0.0
1      0.444444                0.0                0.0                0.0
2      0.111111                0.0                0.0                0.0
3      0.814815                0.0                1.0                0.0
4      0.555556                0.0                1.0                0.0

   education_level_M  education_level_D
0                0.0                0.0
1                0.0                1.0
2                1.0                0.0
3                0.0                0.0
4                0.0                0.0
```

[ ]:
```
df.head(1)
```

[ ]:
```
   age gender occupation education_level marital_status  income  credit_score  \
0   32   Male   Engineer      Bachelor's        Married   85000           720

  loan_status
0    Approved
```

[ ]:
```
X = df_train.drop(['loan_status'], axis=1)
y = df_train['loan_status']

X.shape, y.shape
```

[ ]:
```
((61, 11), (61,))
```

[ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
X_train.shape, X_test.shape
```

[ ]:
```
((48, 11), (13, 11))
```

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

[ ]:
```
     gender  occupation  marital_status       age    income  credit_score  \
3       0.0        16.0             0.0  1.665750  0.516129      0.814815
53      0.0        17.0             0.0  0.827970  0.387097      0.666667
17      0.0        20.0             0.0  0.468921  0.645161      0.888889
8       1.0        14.0             0.0 -0.009810  0.432258      0.703704
6       1.0        15.0             0.0  0.588604  0.612903      0.851852

     education_level_A  education_level_B  education_level_C  \
3                  0.0                1.0                0.0
53                 0.0                0.0                0.0
17                 0.0                0.0                1.0
8                  0.0                0.0                0.0
6                  0.0                0.0                1.0

     education_level_M  education_level_D
3                  0.0                0.0
53                 0.0                1.0
17                 0.0                0.0
8                  0.0                1.0
6                  0.0                0.0
```

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

[ ]:
```
3     0.0
53    0.0
17    0.0
8     0.0
6     0.0
Name: loan_status, dtype: float64
```

[ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

logistic_pipeline = Pipeline([
    ('classifier', LogisticRegression())
])

svm_pipeline = Pipeline([
    ('classifier', SVC())
])

random_forest_pipeline = Pipeline([
    ('classifier', RandomForestClassifier())
])
```

```
logistic_pipeline.fit(X_train, y_train)
svm_pipeline.fit(X_train, y_train)
random_forest_pipeline.fit(X_train, y_train)

logistic_pred = logistic_pipeline.predict(X_test)
svm_pred = svm_pipeline.predict(X_test)
rf_pred = random_forest_pipeline.predict(X_test)

accuracy_score(y_test, logistic_pred), accuracy_score(y_test, svm_pred),␣
↪accuracy_score(y_test, rf_pred)
```

[ ]:
```
(1.0, 0.6923076923076923, 0.9230769230769231)
```

[ ]:
```
from sklearn.model_selection import GridSearchCV

pipeline = Pipeline([
    ('classifier', LogisticRegression())
])

param_grid = [
    {
        'classifier': [LogisticRegression()],
        'classifier__C': [0.1, 1, 10]
    },
    {
        'classifier': [SVC()],
        'classifier__C': [0.1, 1, 10],
        'classifier__kernel': ['linear', 'rbf']
    },
    {
        'classifier': [RandomForestClassifier()],
        'classifier__n_estimators': [100, 200],
        'classifier__max_depth': [10, 20]
    }
]

grid_search = GridSearchCV(pipeline, param_grid, cv=5)
grid_search.fit(X_train, y_train)

best_model = grid_search.best_estimator_

y_pred = best_model.predict(X_test)

accuracy_score(y_test, y_pred), best_model
```

[ ]:
```
(1.0,
 Pipeline(steps=[('classifier',
```

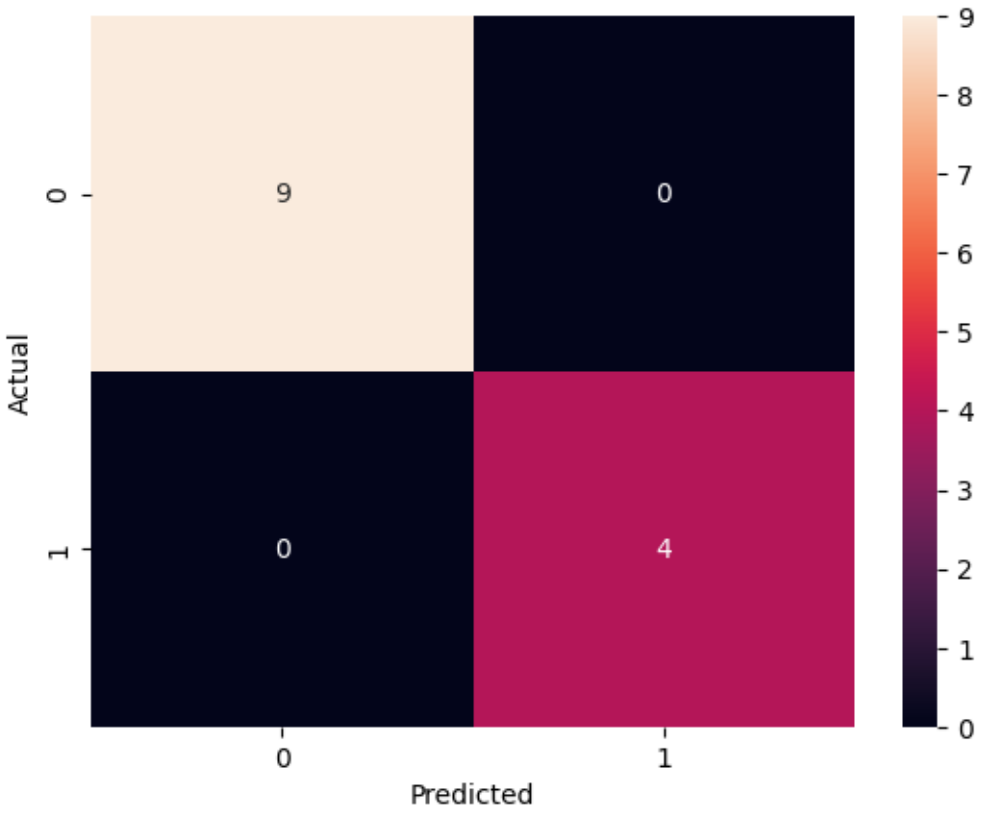
```
                    RandomForestClassifier(max_depth=20, n_estimators=200))]))
```

[ ]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00         9
         1.0       1.00      1.00      1.00         4

    accuracy                           1.00        13
   macro avg       1.00      1.00      1.00        13
weighted avg       1.00      1.00      1.00        13
```

[ ]:
```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))

import seaborn as sns
import matplotlib.pyplot as plt

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
[[9 0]
 [0 4]]
```


9
0
0
4
0
1
Actual
Predicted
9
8
7
6
5
4
3
2
1
0