In [1]:

```
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

from lazypredict.Supervised import LazyClassifier

import warnings
warnings.filterwarnings('ignore')
```

```
/Users/rahulraj/opt/anaconda3/envs/env/lib/python3.8/site-packages/skl
earn/utils/deprecation.py:143: FutureWarning: The sklearn.utils.testin
g module is  deprecated in version 0.22 and will be removed in version
0.24. The corresponding classes / functions should instead be imported
from sklearn.utils. Anything that cannot be imported from sklearn.util
s is now part of the private API.
  warnings.warn(message, FutureWarning)
```

In [ ]:

```

```

In [2]:

```

data = pd.read_csv("credit-card-data.csv")


#data = pd.read_csv("breast-cancer-data.csv")
```

In [3]:

```
data.head(5)
```

Out[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24
0	0	-1.36	-0.07	2.54	1.38	-0.34	0.46	0.24	0.10	0.36	...	-0.02	0.28	-0.11	0.07
1	0	1.19	0.27	0.17	0.45	0.06	-0.08	-0.08	0.09	-0.26	...	-0.23	-0.64	0.10	-0.34
2	1	-1.36	-1.34	1.77	0.38	-0.50	1.80	0.79	0.25	-1.51	...	0.25	0.77	0.91	-0.69
3	1	-0.97	-0.19	1.79	-0.86	-0.01	1.25	0.24	0.38	-1.39	...	-0.11	0.01	-0.19	-1.18
4	2	-1.16	0.88	1.55	0.40	-0.41	0.10	0.59	-0.27	0.82	...	-0.01	0.80	-0.14	0.14

5 rows × 31 columns

In [4]:

```
data.shape
```

Out[4]:

```
(5000, 31)
```

In [5]:

```
data284 = pd.read_csv('credit-card-284k-data.csv')
```

In [6]:

```
data284.shape
```

Out[6]:

```
(284807, 31)
```

In [7]:

```
data.isnull().sum()
```

Out[7]:

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

In [ ]:

```
data.head(2)
```

In [ ]:

In [ ]:

```
#data.isnull().sum()


train , output = 100 , 100

80 , 80 ..training .... xtrain ,ytrain

20 , 20 ... test ...xtest , ytest

1 to 80 ...
```

In [ ]:

In [8]:

```

train = data.iloc[:,:-1]

output = data.iloc[:,30]

scaler = StandardScaler()

xtrain,xtest,ytrain,ytest = train_test_split(train,output,test_size=0.2,random_state

xtrain_scaled = scaler.fit_transform(xtrain)

xtest_scaled = scaler.transform(xtest)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [9]:

```
print(len(xtrain) , len(xtest) , len(ytrain) , len(ytest) )
```

```
4000 1000 4000 1000
```

In [ ]:

```
xtrain
```

In [ ]:

```

```

In [ ]:

```
train.head(2)
```

In [ ]:

```
#test.head(2)
```

In [10]:

```
clf = LazyClassifier(verbose=0,ignore_warnings=True, custom_metric=None)
```

In [11]:

```
models,predictions = clf.fit(xtrain_scaled, xtest_scaled, ytrain, ytest)
```

```
100%|██████████| 29/29 [00:07<00:00,  3.97it/s]
```

In [12]:

```
predictions
```

Out[12]:

	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
Model					
AdaBoostClassifier	1.00	1.00	None	1.00	0.98
LinearDiscriminantAnalysis	1.00	1.00	None	1.00	0.10
XGBClassifier	1.00	1.00	None	1.00	0.32
SVC	1.00	1.00	None	1.00	0.09
SGDClassifier	1.00	1.00	None	1.00	0.05
RidgeClassifierCV	1.00	1.00	None	1.00	0.06
RidgeClassifier	1.00	1.00	None	1.00	0.05
RandomForestClassifier	1.00	1.00	None	1.00	0.80
QuadraticDiscriminantAnalysis	1.00	1.00	None	1.00	0.06
PassiveAggressiveClassifier	1.00	1.00	None	1.00	0.04
LogisticRegression	1.00	1.00	None	1.00	0.08
BaggingClassifier	1.00	1.00	None	1.00	0.21
LinearSVC	1.00	1.00	None	1.00	0.08
LabelSpreading	1.00	1.00	None	1.00	1.60
LabelPropagation	1.00	1.00	None	1.00	0.86
KNeighborsClassifier	1.00	1.00	None	1.00	0.41
ExtraTreesClassifier	1.00	1.00	None	1.00	0.34
ExtraTreeClassifier	1.00	1.00	None	1.00	0.05
CalibratedClassifierCV	1.00	1.00	None	1.00	0.18
BernoulliNB	1.00	1.00	None	1.00	0.04
LGBMClassifier	1.00	1.00	None	1.00	0.54
NearestCentroid	1.00	1.00	None	1.00	0.03
DummyClassifier	1.00	1.00	None	1.00	0.07
Perceptron	1.00	1.00	None	1.00	0.05
DecisionTreeClassifier	1.00	1.00	None	1.00	0.10
GaussianNB	1.00	1.00	None	1.00	0.04

In [ ]:

```
ML = 10  ...80
DL = 10 ....80


20 ..90
20 ...90

100 ...92
100 ...92





1000000 ....92
1000000000000000 ...99.12334
```

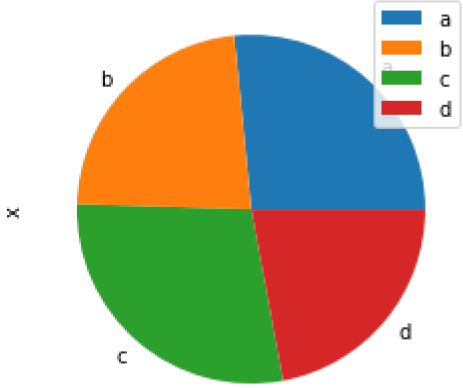
In [13]:

```
df = pd.DataFrame(3 * np.random.rand(4), index=['a', 'b', 'c', 'd'],columns=['x'])


df.plot.pie(subplots=True)
```

Out[13]:

```
array([<AxesSubplot:ylabel='x'>], dtype=object)
```



In [14]:

```
print (pd.datetime.now())
```

```
2021-05-20 16:42:49.785555
```

In [15]:

```
print (pd.date_range('5/10/2021', periods=10))
```

```
DatetimeIndex(['2021-05-10', '2021-05-11', '2021-05-12', '2021-05-13',
               '2021-05-14', '2021-05-15', '2021-05-16', '2021-05-17',
               '2021-05-18', '2021-05-19'],
              dtype='datetime64[ns]', freq='D')
```

In [16]:

```
print (pd.bdate_range('5/10/2021', periods=15))
```

```
DatetimeIndex(['2021-05-10', '2021-05-11', '2021-05-12', '2021-05-13',
               '2021-05-14', '2021-05-17', '2021-05-18', '2021-05-19',
               '2021-05-20', '2021-05-21', '2021-05-24', '2021-05-25',
               '2021-05-26', '2021-05-27', '2021-05-28'],
              dtype='datetime64[ns]', freq='B')
```

In [ ]:

```

```

In [ ]:

```
2,3,4,5,6,7,100,120 ... ... 0-1
```

In [17]:

```
from sklearn.preprocessing import StandardScaler
import numpy as np
import seaborn as sns
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np
from lazypredict.Supervised import LazyRegressor
```

In [19]:

```
df = pd.read_csv('house-price-data.csv')
```

In [20]:

```
df.head(2)
```

Out[20]:

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Ut
0	1	60	RL	65.00	8450	Pave	NaN	Reg	Lvl	A
1	2	20	RL	80.00	9600	Pave	NaN	Reg	Lvl	A

2 rows × 81 columns

In [21]:

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

Out[21]:

```
Normal     1198
Partial     125
Abnorml     101
Family       20
Alloca       12
AdjLand       4
Name: SaleCondition, dtype: int64
```

In [ ]:

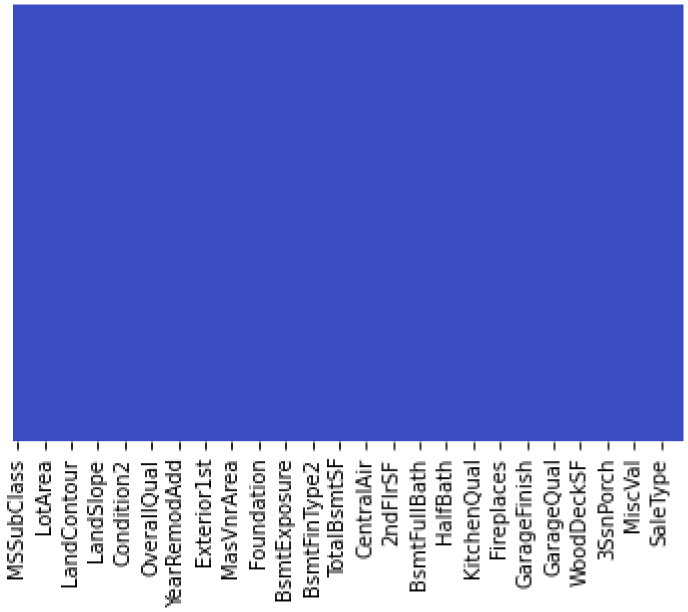
In [ ]:

In [22]:

```
df['LotFrontage'] = df['LotFrontage'].fillna(df['LotFrontage'].mean())
df.drop(['Alley'],axis=1,inplace=True)
df['BsmtCond'] = df['BsmtCond'].fillna(df['BsmtCond'].mode()[0])
df['BsmtQual'] = df['BsmtQual'].fillna(df['BsmtQual'].mode()[0])
df['FireplaceQu'] = df['FireplaceQu'].fillna(df['FireplaceQu'].mode()[0])
df['GarageType'] = df['GarageType'].fillna(df['GarageType'].mode()[0])
df.drop(['GarageYrBlt'],axis=1,inplace=True)
df['GarageFinish'] = df['GarageFinish'].fillna(df['GarageFinish'].mode()[0])
df['GarageQual'] = df['GarageQual'].fillna(df['GarageQual'].mode()[0])
df['GarageCond'] = df['GarageCond'].fillna(df['GarageCond'].mode()[0])
df.drop(['PoolQC','Fence','MiscFeature'],axis=1,inplace=True)
df.drop(['Id'],axis=1,inplace=True)
df['MasVnrType'] = df['MasVnrType'].fillna(df['MasVnrType'].mode()[0])
df['MasVnrArea'] = df['MasVnrArea'].fillna(df['MasVnrArea'].mode()[0])
df['BsmtExposure'] = df['BsmtExposure'].fillna(df['BsmtExposure'].mode()[0])
df['BsmtExposure'] = df['BsmtExposure'].fillna(df['BsmtExposure'].mode()[0])
df['BsmtExposure'] = df['BsmtExposure'].fillna(df['BsmtExposure'].mode()[0])
df['BsmtFinType1'] = df['BsmtFinType1'].fillna(df['BsmtFinType1'].mode()[0])
df.dropna(inplace=True)
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='coolwarm')
```

Out[22]:

```
<AxesSubplot:>
```



In [23]:

```
numeric = list(df.dtypes[df.dtypes!='object'].index)
train = df[numeric]
```

In [ ]:

```
train.head(5)
```

In [24]:

```
price = train.iloc[:,35:36]
train = train.iloc[:,:-1]

xtrain,xtest,ytrain,ytest = train_test_split(train,
                                             price,
                                             test_size=0.1,
                                             random_state=40)
```

In [27]:

```
scaler = StandardScaler()
xtrain_scaled = scaler.fit_transform(xtrain)

xtest_scaled = scaler.transform(xtest)

clf = LazyRegressor(verbose=0,ignore_warnings=False, custom_metric=None)


models,predictions = clf.fit(xtrain_scaled, xtest_scaled, ytrain, ytest)
```

```
100%|██████████| 42/42 [00:09<00:00,  4.21it/s]
```

In [28]:

```
predictions
```

Out[28]:

	Adjusted R-Squared	R-Squared	RMSE	Time Taken
Model				
GradientBoostingRegressor	0.87	0.90	22757.70	0.50
XGBRegressor	0.87	0.90	22861.33	0.29
RandomForestRegressor	0.86	0.89	23882.74	2.16
LGBMRegressor	0.85	0.89	23985.34	0.24
HistGradientBoostingRegressor	0.85	0.89	24084.11	1.30
ExtraTreesRegressor	0.85	0.89	24373.57	0.95
BaggingRegressor	0.84	0.88	25468.82	0.16
HuberRegressor	0.84	0.88	25504.16	0.08
LassoLarsIC	0.83	0.88	25574.99	0.04
PassiveAggressiveRegressor	0.83	0.87	25740.43	0.12
PoissonRegressor	0.83	0.87	25791.42	0.05
Lars	0.82	0.87	26439.02	0.04
BayesianRidge	0.82	0.86	26752.53	0.03
LassoLars	0.82	0.86	26764.13	0.04
TransformedTargetRegressor	0.82	0.86	26794.09	0.02
LinearRegression	0.82	0.86	26794.09	0.03
RidgeCV	0.82	0.86	26804.48	0.07
Ridge	0.82	0.86	26837.43	0.06
Lasso	0.82	0.86	26839.52	0.04
LassoLarsCV	0.82	0.86	26871.64	0.10
LassoCV	0.82	0.86	26982.30	0.13
LarsCV	0.82	0.86	26987.85	0.09
SGDRegressor	0.81	0.86	27294.74	0.03
ElasticNet	0.81	0.86	27495.16	0.08
AdaBoostRegressor	0.80	0.85	27963.70	0.36
GammaRegressor	0.80	0.85	28042.93	0.02
TweedieRegressor	0.79	0.84	28972.53	0.04
GeneralizedLinearRegressor	0.79	0.84	28972.53	0.03
RANSACRegressor	0.78	0.83	29618.27	0.15
OrthogonalMatchingPursuitCV	0.77	0.82	30365.37	0.04
OrthogonalMatchingPursuit	0.74	0.81	31932.00	0.03
KNeighborsRegressor	0.66	0.74	36949.89	0.05
ExtraTreeRegressor	0.64	0.73	37582.75	0.03

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
DecisionTreeRegressor	0.61	0.71	39055.91	0.08
ElasticNetCV	-0.13	0.15	66733.46	0.21
DummyRegressor	-0.33	-0.00	72477.31	0.07
NuSVR	-0.35	-0.01	73001.30	0.10
SVR	-0.40	-0.06	74466.37	0.21
GaussianProcessRegressor	-6.57	-4.71	173144.04	0.24
KernelRidge	-7.69	-5.54	185411.14	0.13
MLPRegressor	-8.51	-6.16	193966.50	1.49
LinearSVR	-8.61	-6.24	195001.70	0.03

In [ ]:

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

In [ ]:

```
y = pd.get_dummies(df['BsmtCond'])
```

In [ ]:

```
df['BsmtCond'].head(5)
```

In [ ]:

```
df['BsmtCond'].head(5)
```

In [ ]:

```
y.head(5)
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
```

In [ ]:

```
label = LabelEncoder()

x = label.fit_transform(df['GarageQual'])
```

In [ ]:

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

In [ ]:

```
for i in x:
    print(i)
```

In [ ]:

```
data = pd.get_dummies(df['GarageQual'])
```

In [ ]:

```
data.head(2)
```

In [ ]:

```

```

In [ ]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.ensemble import AdaBoostClassifier


pd.set_option("display.max_rows", None, "display.max_columns", None)

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

import warnings
warnings.filterwarnings('ignore')
```

In [ ]:

```

```

In [ ]:

```
ada = AdaBoostClassifier(n_estimators=5, random_state=0)
```

In [ ]:

```
ada.fit(xtrain_scaled,ytrain)
```

In [ ]:

```
ada.score(xtest_scaled,ytest)
```

In [ ]:

```

```

In [ ]:

```

```