Pattern & Anomaly detection Lab

Exp 5

Dhruv Singhal || 500075346 || R177219074 || AIML B3 || Sem5

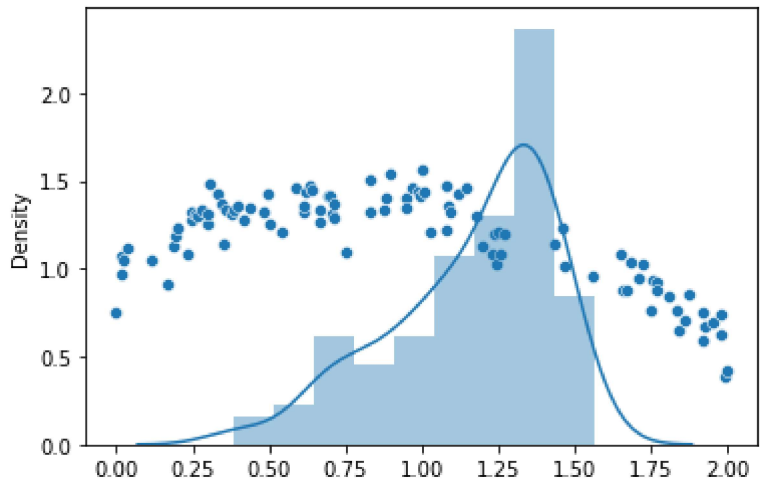
In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
n_samples = 100
de_linearize = lambda X: np.sin(X) + np.cos(X )
X1 = np.sort(np.random.rand(n_samples)) * 2
X2= np.sort(np.random.rand(n_samples)) * 2
X3 = np.sort(np.random.rand(n_samples)) * 2
X4 = np.sort(np.random.rand(n_samples)) * 2
X5 = np.sort(np.random.rand(n_samples)) * 2
X=[]
feat=[]
for i in range(0,n_samples):
    feat.append([X1[i],X2[i],X3[i],X4[i],X5[i]])
feats=pd.DataFrame(feat,columns=('F1','F2','F3','f4','F5'))
print(feats.head())
for i in range(0,n_samples):
    a=(X1[i]+X2[i]+X3[i]+X4[i]+X5[i])/5
    X.append(a)
y = de_linearize(X) + np.random.randn(n_samples) * 0.1
sns.distplot(y)
sns.scatterplot(X1,y)
```

```
         F1        F2        F3        f4        F5
0  0.000030  0.002443  0.027655  0.030982  0.016047
1  0.017742  0.028494  0.040273  0.045429  0.024450
2  0.019404  0.038491  0.041325  0.048317  0.029770
3  0.021556  0.042666  0.058933  0.057721  0.045747
4  0.037288  0.074147  0.077952  0.062938  0.052777
```

Out[2]:

```
<AxesSubplot:ylabel='Density'>
```



In [3]:

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
poly = PolynomialFeatures(degree = 2)
X_poly = poly.fit_transform(feats)
```

In [4]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X_poly,y,test_size=0.20,random_state=11)
lin = LinearRegression()
lin.fit(X_train, Y_train)
y_pred=lin.predict(X_test)
```

In [5]:

```
from sklearn.linear_model import Ridge
rid=Ridge()
model=rid.fit(X_train,Y_train,sample_weight=None)
y_pred2=model.predict(X_test)
```

In [6]:

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
kfold = KFold(n_splits=5, random_state=11, shuffle=True)
```

In [7]:

```
models = LinearRegression()
scores = cross_val_score(models, X_poly, y, scoring='neg_mean_absolute_error',cv=kfold, n_jobs=-1)
#scores.fit(X_train,Y_train)
np.mean(np.absolute(scores))
```

Out[7]:

```
0.09303971371329463
```

In [8]:

```
import sklearn.metrics
print("Score For Linear regression(Polynomial) is:")
print("R^2 Score",sklearn.metrics.r2_score(Y_test, y_pred))
print("Mean Squared Log Error",sklearn.metrics.mean_squared_log_error(Y_test, y_pred))
print("Mean Squared Error",sklearn.metrics.mean_squared_error(Y_test, y_pred, squared=True))
print("train Accuracy:",lin.score(X_train,Y_train))
print("test Accuracy:",lin.score(X_test,Y_test))
```

```
Score For Linear regression(Polynomial) is:
R^2 Score 0.7996060278329573
Mean Squared Log Error 0.004779086806566104
Mean Squared Error 0.01978215050160775
train Accuracy: 0.924503960054819
test Accuracy: 0.7996060278329573
```

In [9]:

```
print("Score For Ridge is:")
print("R^2 Score",sklearn.metrics.r2_score(Y_test, y_pred2))
print("Mean Squared Log Error",sklearn.metrics.mean_squared_log_error(Y_test, y_pred2))
print("Mean Squared Error",sklearn.metrics.mean_squared_error(Y_test, y_pred2, squared=True))
```

```
print("train Accuracy:",model.score(X_train,Y_train))
print("test Accuracy:",model.score(X_test,Y_test))
```

```
Score For Ridge is:
R^2 Score 0.8583910485406168
Mean Squared Log Error 0.0037409478680416964
Mean Squared Error 0.01397911104735863
train Accuracy: 0.8412770171525001
test Accuracy: 0.8583910485406168
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: