Mohsensho copy file with no result History

0 contributors

577 lines (577 sloc) | 12.3 KB

In [ ]:
```
# Importing library
import pandas as pd # data processing, CV file I/0 (e.g. pd.read csv)
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import preprocessing # Import Label Encoder
```

In [ ]:
```
# Read csv
train_df = pd.read_csv('Datasets/Mercedes_train.csv')
test_df = pd.read_csv('Datasets/Mercedes_test.csv')
print (train_df.shape) # Find Number of rows and columns
print (train_df.columns)
print (test_df.shape) # Find Number of rows and columns
print (test_df.columns)
train_df.head()
# Show first 5 records
```

In [ ]:
```
# Describe the dataset i.r.t its data Distribution
train_df.describe()
```

In [ ]:
```
#If for any column(s), the variance is equal to zero, then you need to remove those variable(s
# Check the variance
train_df.var(numeric_only=True)
```

In [ ]:
```
# Find out the variance is equal to zero for any columns
(train_df.var(numeric_only=True) == 0)
```

In [ ]:
```
(train_df.var(numeric_only=True) == 0).values
```

In [ ]:
```
variance_with_zero = train_df.var(numeric_only=True)[train_df.var(numeric_only=True)==0].index
variance_with_zero
```

In [ ]:
```
# Drop zero variance variables
train_df = train_df.drop(variance_with_zero, axis=1)
```

In [ ]:
```
print(train_df.shape)
```

In [ ]:
```
# As ID column is irrelevant for our prediction hence we drop this column
train_df = train_df.drop(('ID'), axis=1)
```

In [ ]:
```
train_df.head()
```

In [ ]:
```
#Check for null and unique values for test and train sets.
train_df.isnull().sum().values
```

In [ ]:
```
train_df.isnull().any()
```

In [ ]:
```
test_df.isnull().sum().values
```

In [ ]:
```
# Find unique records
train_df.nunique()
```

In [ ]:
```
#Filter out the columns having object datatype
```

```
#Filter out the columns having object datatype
object_datatypes = train_df.select_dtypes(include=[object])
object_datatypes
```

In [ ]:
```
object_datatype_columns = object_datatypes.columns
object_datatype_columns
```

In [ ]:
```
#Apply label encoder.
# Initialize Label Encoder object
label_encoder = preprocessing.LabelEncoder()
train_df['X0'].unique()
```

In [ ]:
```
# Encode and transform object data to interger
train_df['X0'] = label_encoder.fit_transform(train_df['X0'])
```

In [ ]:
```
train_df['X0'].unique()
```

In [ ]:
```
# Apply same for all columns having object type data
train_df['X1'] = label_encoder.fit_transform(train_df['X1'])
train_df['X2'] = label_encoder.fit_transform(train_df['X2'])
train_df['X3'] = label_encoder.fit_transform(train_df['X2'])
train_df['X4'] = label_encoder.fit_transform(train_df['X3'])
train_df['X5'] = label_encoder.fit_transform(train_df['X4'])
train_df['X6'] = label_encoder.fit_transform(train_df['X5'])
train_df['X8'] = label_encoder.fit_transform(train_df['X8'])
```

In [ ]:
```
train_df.head()
```

In [ ]:
```
#Perform dimensionality reduction (PCA)
from sklearn.decomposition import PCA
# PCA with 958
sklearn_pca = PCA(n_components=0.95)

sklearn_pca.fit(train_df)
```

In [ ]:
```
x_train_transformed = sklearn_pca.transform(train_df)
```

In [ ]:
```
print(x_train_transformed.shape)
```

In [ ]:
```
# PCA with 988
sklearn_pca_98 = PCA(n_components=0.98)
```

In [ ]:
```
sklearn_pca_98.fit(train_df)
```

In [ ]:
```
x_train_transformed_98 = sklearn_pca_98.transform(train_df)
print(x_train_transformed_98.shape)
```

In [ ]:
```
train_df.y
```

In [ ]:
```
# Train and Test split on Train dataset
X = train_df.drop('y', axis=1)
y = train_df.y
xtrain,xtest,ytrain,ytest = train_test_split(X,y, test_size=0.3, random_state=42)
```

In [ ]:
```
print('train')
```

```
print (xtrain)
print (xtrain.shape)
```

In [ ]:
```
print(ytrain)
print(ytrain.shape)
```

In [ ]:
```
print (xtest)
print (xtest.shape)
```

In [ ]:
```
# PCA with 958 for xtrain
pca_xtrain = PCA(n_components=0.95)
pca_xtrain.fit(xtrain)
```

In [ ]:
```
pca_xtrain_transformed = pca_xtrain.transform(xtrain)
print(pca_xtrain_transformed.shape)
```

In [ ]:
```
#PCA with 958 for xtest
pca_xtest = PCA(n_components=0.95)
pca_xtest.fit(xtest)
```

In [ ]:
```
pca_xtest_transformed = pca_xtest.transform(xtest)
print (pca_xtest_transformed.shape)
```

In [ ]:
```
print (pca_xtest.explained_variance_)
print (pca_xtest.explained_variance_ratio_)
```

In [ ]:
```
#PCA for test df dataset
test_df
```

In [ ]:
```
test_object_datatypes = test_df.select_dtypes(include=[object])
test_object_datatypes
```

In [ ]:
```
test_df['X0'] = label_encoder.fit_transform(test_df['X0'])
test_df['X1'] = label_encoder.fit_transform(test_df['X1'])
test_df['X2'] = label_encoder.fit_transform(test_df['X2'])
test_df['X3'] = label_encoder.fit_transform(test_df['X3'])
test_df['X4'] = label_encoder.fit_transform(test_df['X4'])
test_df['X5'] = label_encoder.fit_transform(test_df['X5'])
test_df['X6'] = label_encoder.fit_transform(test_df['X6'])
test_df['X8'] = label_encoder.fit_transform(test_df['X8'])
```

In [ ]:
```
print(test_df)
print(test_df.shape)
```

In [ ]:
```
test_df = test_df.drop('ID',axis=1)
```

In [ ]:
```
# PCA with 958 for test df
pca_test_df = PCA(n_components=0.95)
pca_test_df.fit(test_df)
```

In [ ]:
```
pca_test_df_transformed = pca_test_df.transform(test_df)
print (pca_test_df_transformed.shape)
```

In [ ]:
```
print(pca_test_df.explained_variance_)
```

```
print(pca_test_df.explained_variance_ratio_)
```

In [ ]:
```
y
```

In [ ]:
```
#Perform XGboost
from sklearn import svm
from sklearn import model_selection
import xgboost as xgb
```

In [ ]:
```
from xgboost import XGBRegressor
xgb = XGBRegressor()
xgb.fit(xtrain, ytrain)
```

In [ ]:
```
pred = xgb.predict(xtest)
pred
```

In [ ]:
```
df_res = pd.DataFrame(pred, columns = ["yHat"])
df_res
```

In [ ]:
```
df_res.to_csv('submission.csv',index=False)
```