```
For cat variable : catcol = list(data.select_dtypes(exclude ='number').columns)
Value counting: data['department'].value_counts() or  data.wip.isnull().sum()
Replacing: data.department.replace('finishing ', "finishing", inplace = True)
Drop target: Y = data['actual_productivity'];x =data.drop(['actual_productivity'],axis=1)
x_train.GDP.mean(); simpleimp:X_train['income'] = si.fit_transform(X_train[['income']])
income_median = X_train['income'].median();X_train.fillna(value= {"income":
income_median},inplace=True);from scipy import stats stats.mode(X_train['policy'])[0][0])
Poly features: poly1 = PolynomialFeatures(degree=2)
X_train_poly = poly1.fit_transform(X_train[X_train.columns[2:5]]) some selected column
Transformer &Pipeline: catfeat = ['date', 'department'];numfeat = ['team', 'smv', 'wip']
num_tran = pipeline(
steps=[('simimp',SimpleImputer(missing_values=np.nan,strategy='mean'),('stdscl',StandardS
caler()))]);  cat_tran = OneHotEncoder(handle_unknown='ignore')
from sklearn.compose import ColumnTransformer
process = ColumnTransformer
(transformers=[('num',num_tran,numfeat),('cat',cat_tran,catfeat)])
estimator = LinearRegression();selector = RFE(estimator, n_features_to_select=5,step=1)
selector.support_,   selector.ranking_
kf = KFold(n_splits=5,random_state=42,shuffle=True);sfs = SequentialFeatureSelector(lr,
n_features_to_select=5,cv=kf,direction='forward'); sfs.fit_transform(pro_X_train,
y_train) print(sfs.get_support(indices=True))
Score:np.round(lr.score(pro_X_test,y_test),4)
Grid Search:grid_pipeline = Pipeline([("poly",PolynomialFeatures()),("Lasso", Lasso())])
param_grid ={'poly__degree': (1, 2), 'Lasso__alpha': np.logspace(-3, 0, num=5)}
Lasso_grid_search = GridSearchCV(grid_pipeline, param_grid=param_grid,
scoring="neg_mean_absolute_error", return_train_score=False)
Lasso_grid_search.fit(pro_X_train,y_train) Lasso_grid_search.best_params_
PCA: from sklearn.decomposition import PCA
pca = PCA(n_components=5,whiten=True,svd_solver='full', random_state=42); X_train_PCA =
pca.fit_transform(pro_X_train)
Plot: data.corr();plt.figure(figsize=(12,8));sns.heatmap(data.corr(), annot=True)
sns.set_theme(style='whitegrid');ax= sns.violinplot(x=X['incentive'])
print("samples correspond to class 0:",np.sum(y==0))
df = pd.DataFrame(x, columns = data1.feature_names) convert np to pd
from sklearn.feature_selection import mutual_info_regression , SelectPercentile
mir = SelectPercentile(mutual_info_regression, percentile=10)
chd1 = mir.fit_transform(chd, cht)
print(chd1.shape)
```

use the strategy 'mean' to predict all the labels. Calculate the coefficient of determination R 2 of the prediction.

```
from sklearn.metrics import r2_score
model = LinearRegression(); mean_val = np.mean(y_train)
model.fit(x_train,np.full_like(y_train, mean_val)); y_pred = model.predict(x_test)
r2 = r2_score(y_test,y_pred)
```

Calculate the mean absolute error for **training data**.

```
scal = MinMaxScaler();X_train_new = scal.fit_transform(x_train)
X_test_new = scal.transform(x_test);model = SGDRegressor();
```

```
model.fit(X_train_new, y_train);y_train_pred = model.predict(X_train_new)
meanerr = mean_absolute_error(y_train, y_train_pred)
```

mean absolute error for **test data.**

```
y_test_pred = model.predict(X_test_new);
meanerrtest = mean_absolute_error(y_test, y_test_pred)
```

print the scores that are calculated using **5 fold cross validation using LinearRegression** on entire dataset.

```
X,Y = data.data , data.target ;model = LinearRegression();
cvs = cross_val_score(model, X,Y,cv = 5);print(cvs)
```