```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

```
train.head()
```

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	X12	X13	X14	X15	X16	X
0	0	130.81	k	v	at	a	d	u	j	o	0	0	0	1	0	0	0	
1	6	88.53	k	t	av	e	d	y	l	o	0	0	0	0	0	0	0	
2	7	76.26	az	w	n	c	d	x	j	x	0	0	0	0	0	0	0	
3	9	80.62	az	t	n	f	d	x	l	e	0	0	0	0	0	0	0	
4	13	78.02	az	v	n	f	d	h	d	n	0	0	0	0	0	0	0	

5 rows × 378 columns

```
test.head()
```

```
print('Size of training set: {} rows and {} columns'.format(*train.shape))
print('Size of testing set: {} rows and {} columns'.format(*test.shape))
```

```
Size of training set: 4209 rows and 378 columns
Size of testing set: 4209 rows and 377 columns
```

```
cols = [c for c in train.columns if 'X' in c]
print('Number of features: {}'.format(len(cols)))
print('Feature types:')
train[cols].dtypes.value_counts()
```

```
Number of features: 376
Feature types:
int64     368
object      8
dtype: int64
```

Check for null and unique values

```
usable_columns = list(set(train.columns) - set(['ID', 'y']))
y_train = train['y'].values
id_test = test['ID'].values

x_train = train[usable_columns]
x_test = test[usable_columns]
```

```
def check_missing_values(df):
    if df.isnull().any().any():
        print('There are missing values in the dataframe')
    else:
        print('There are no missing values in the dataframe')
```

```
check_missing_values(x_train)
check_missing_values(x_test)
```

```
There are no missing values in the dataframe
There are no missing values in the dataframe
```

label encoder

```
#Label Encoding the categorical values
for column in usable_columns:
    cardinality = len(np.unique(x_train[column]))
    if cardinality == 1:
```

```
        x_train.drop(column, axis=1)
        # value is useless so we drop it
        x_test.drop(column, axis=1)
    if cardinality > 2: # Column is categorical
        mapper = lambda x: sum([ord(digit) for digit in x])
        x_train[column] = x_train[column].apply(mapper)
        x_test[column] = x_test[column].apply(mapper)
x_train.head()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:10: SettingWithCopyWa
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable
  # Remove the CWD from sys.path while we load stuff.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:11: SettingWithCopyWa
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable
  # This is added back by InteractiveShellApp.init_path()
```

	X22	X27	X249	X79	X314	X285	X52	X151	X275	X10	X244	X197	X274	X69	X1
0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	
1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	
2	0	1	0	0	0	0	0	0	0	0	1	0	1	0	
3	0	1	0	0	0	0	0	0	0	0	1	0	0	0	
4	0	1	0	0	0	0	0	0	0	0	0	0	0	0	

5 rows × 376 columns

```
print('Feature:')
x_train[cols].dtypes.value_counts()
```

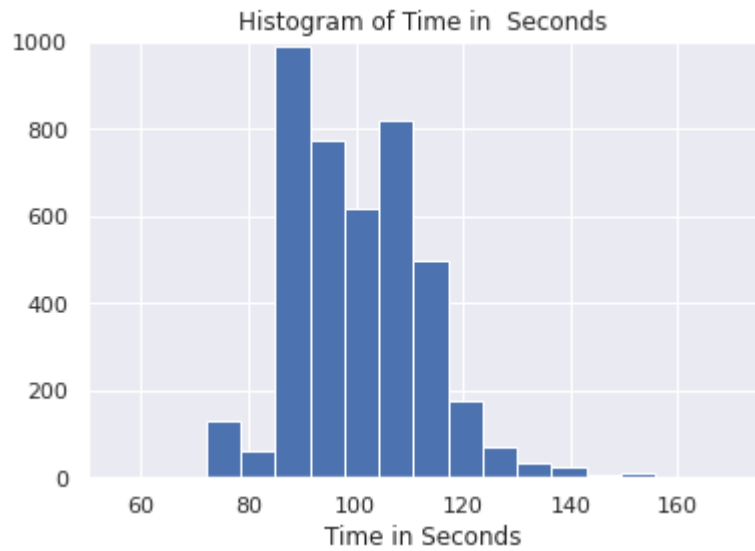
```
Feature:
int64    376
dtype: int64
```

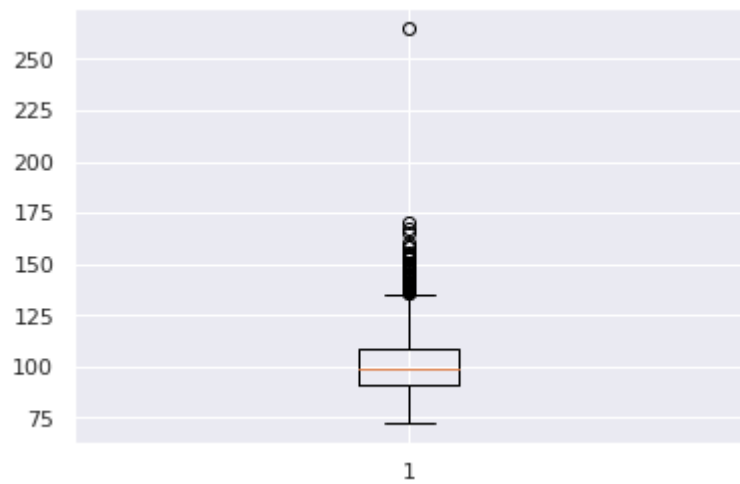
```
n_comp = 15
pca = PCA(n_components = n_comp,random_state = 425)
pca2_results_train = pca.fit_transform(x_train)
pca2_results_test = pca.transform(x_test)
```

```
plt.hist(train.y, bins = 30)
plt.xlabel('Time in Seconds')
plt.title('Histogram of Time in  Seconds')
```

```
plt.title('Histogram of Time in  Seconds')
plt.grid(True)
plt.axis([50,175, 0,1000])
plt.show()

p = plt.boxplot(train.y)
plt.show()
```





XGBoost

```
import xgboost as xgb
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

```
x_train,x_val,y_train,y_val = train_test_split(pca2_results_train, y_train, test_size=0.2, ra
```

```
d_train = xgb.DMatrix(x_train,label = y_train)
d_val = xgb.DMatrix(x_val,label = y_val)
d_test = xgb.DMatrix(pca2_results_test)
```

```
params = {}
params['objective'] = 'reg:linear'
params['eta'] = 0.02
params['max_depth'] = 4

def xgb_r2_score(preds, dtrain):
    labels = dtrain.get_label()
    return 'r2', r2_score(labels, preds)
watchlist = [(d_train, 'train'), (d_val, 'valid')]
clf = xgb.train(params, d_train,  1000, watchlist, early_stopping_rounds=50,
 feval=xgb_r2_score, maximize=True, verbose_eval=10)
```

```
Multiple eval metrics have been passed: 'valid-r2' will be used for early stopping.

Will train until valid-r2 hasn't improved in 50 rounds.
[10]    train-rmse:81.2761      valid-rmse:80.364       train-r2:-38.8838       valid
[20]    train-rmse:66.7133      valid-rmse:65.7777      train-r2:-25.8718       valid
[30]    train-rmse:54.8646      valid-rmse:53.8977      train-r2:-17.1742       valid
[40]    train-rmse:45.2369      valid-rmse:44.2292      train-r2:-11.3554       valid
[50]    train-rmse:37.4325      valid-rmse:36.3737      train-r2:-7.45996       valid
[60]    train-rmse:31.1228      valid-rmse:30.0205      train-r2:-4.8483        valid
[70]    train-rmse:26.0519      valid-rmse:24.9005      train-r2:-3.09778       valid
[80]    train-rmse:21.9913      valid-rmse:20.8068      train-r2:-1.91994       valid
[90]    train-rmse:18.7784      valid-rmse:17.5693      train-r2:-1.12907       valid
[100]   train-rmse:16.2555      valid-rmse:15.0353      train-r2:-0.595408      valid
[110]   train-rmse:14.3023      valid-rmse:13.0836      train-r2:-0.23504       valid
[120]   train-rmse:12.8008      valid-rmse:11.6028      train-r2:0.010655       valid
[130]   train-rmse:11.6706      valid-rmse:10.5153      train-r2:0.177648       valid
[140]   train-rmse:10.8303      valid-rmse:9.73827      train-r2:0.29181        valid
[150]   train-rmse:10.2101      valid-rmse:9.19158      train-r2:0.370598       valid
[160]   train-rmse:9.74076      valid-rmse:8.80645      train-r2:0.427128       valid
[170]   train-rmse:9.38997      valid-rmse:8.55078      train-r2:0.467647       valid
[180]   train-rmse:9.11758      valid-rmse:8.38015      train-r2:0.498085       valid
[190]   train-rmse:8.9127       valid-rmse:8.27028      train-r2:0.520387       valid
[200]   train-rmse:8.75419      valid-rmse:8.20496      train-r2:0.537296       valid
[210]   train-rmse:8.62817      valid-rmse:8.15833      train-r2:0.550522       valid
[220]   train-rmse:8.51687      valid-rmse:8.12397      train-r2:0.562042       valid
[230]   train-rmse:8.443        valid-rmse:8.10473      train-r2:0.569607       valid
[240]   train-rmse:8.38042      valid-rmse:8.09389      train-r2:0.575964       valid
[250]   train-rmse:8.31392      valid-rmse:8.08616      train-r2:0.582666       valid
[260]   train-rmse:8.25848      valid-rmse:8.08218      train-r2:0.588214       valid
[270]   train-rmse:8.20547      valid-rmse:8.07973      train-r2:0.593483       valid
[280]   train-rmse:8.16457      valid-rmse:8.08421      train-r2:0.597526       valid
[290]   train-rmse:8.12196      valid-rmse:8.08091      train-r2:0.601716       valid
[300]   train-rmse:8.08773      valid-rmse:8.08092      train-r2:0.605066       valid
[310]   train-rmse:8.04613      valid-rmse:8.0808       train-r2:0.609118       valid
[320]   train-rmse:8.0078       valid-rmse:8.08074      train-r2:0.612834       valid
[330]   train-rmse:7.97502      valid-rmse:8.07613      train-r2:0.615996       valid
[340]   train-rmse:7.94557      valid-rmse:8.07635      train-r2:0.618827       valid
[350]   train-rmse:7.91485      valid-rmse:8.0752       train-r2:0.621769       valid
[360]   train-rmse:7.87902      valid-rmse:8.076        train-r2:0.625186       valid

[370]   train-rmse:7.84825      valid-rmse:8.07563      train-r2:0.628108       valid
[380]   train-rmse:7.82156      valid-rmse:8.07338      train-r2:0.630633       valid
```

```
[390]	train-rmse:7.79282	valid-rmse:8.0721	train-r2:0.633343	valid
[400]	train-rmse:7.76938	valid-rmse:8.07165	train-r2:0.635545	valid
[410]	train-rmse:7.7379	valid-rmse:8.06847	train-r2:0.638492	valid
[420]	train-rmse:7.7116	valid-rmse:8.06529	train-r2:0.640945	valid
[430]	train-rmse:7.68462	valid-rmse:8.0634	train-r2:0.643453	valid
[440]	train-rmse:7.65497	valid-rmse:8.0593	train-r2:0.6462 valid-r2:0.53
[450]	train-rmse:7.63564	valid-rmse:8.05814	train-r2:0.647984	valid
[460]	train-rmse:7.60972	valid-rmse:8.05692	train-r2:0.65037	valid
[470]	train-rmse:7.58899	valid-rmse:8.05492	train-r2:0.652272	valid
[480]	train-rmse:7.56926	valid-rmse:8.05578	train-r2:0.654078	valid
[490]	train-rmse:7.54701	valid-rmse:8.05774	train-r2:0.656108	valid
[500]	train-rmse:7.52036	valid-rmse:8.05891	train-r2:0.658533	valid
[510]	train-rmse:7.49104	valid-rmse:8.05584	train-r2:0.661191	valid
[520]	train-rmse:7.47503	valid-rmse:8.0581	train-r2:0.662637	valid
Stopping. Best iteration:
[471]	train-rmse:7.58446	valid-rmse:8.05409	train-r2:0.652688	valid
```

Predict test_df using XGBoost

```
p_test = clf.predict(d_test)
```

```
sub = pd.DataFrame()
sub['ID'] = id_test
sub['y'] = p_test
sub.to_csv('test_df.csv', index = False)
sub.head()
```

	ID	y
0	1	79.169426
1	2	94.651009
2	3	79.888870
3	4	76.194939
4	5	112.719109

✓ 0s completed at 9:17 AM