0. EDA
1. Run Linear Regression
2. Run SVR
3. PCA

In [1]:
```python
import pandas as pd
import seaborn as sns
from sklearn.svm import SVR
import numpy as np
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
# Import PCA
from sklearn import decomposition
from sklearn.decomposition import PCA
from sklearn.preprocessing import scale
from sklearn import linear_model
from sklearn import metrics
from yellowbrick.regressor import ResidualsPlot
import matplotlib.pyplot as plt
```

In [2]:
```python
## Load the data
path="C:\\Users\\fbaharkoush\\IE 598 Machine Learning\\Homework\\HW 5\\"
df_tbd=pd.read_csv(path+"hw5_treasury yield curve data.csv")
df_tbd.drop("Date",axis=1,inplace=True)
df_tbd.shape
```

Out[2]:
```
(8353, 31)
```

In [3]:
```python
df_tbd.describe()
```

Out[3]:

	SVENF01	SVENF02	SVENF03	SVENF04	SVENF05	SVENF06	SVENF07	SVENF08	SVENF09	SVENF10	...	SVENF22	S
count	8353.000000	8353.000000	8353.000000	8353.000000	8353.00000	8353.000000	8353.000000	8353.000000	8353.000000	8353.000000	...	8353.000000	8353
mean	3.895104	4.371348	4.779336	5.128279	5.42020	5.657948	5.845959	5.989599	6.094526	6.166257	...	5.808739	5
std	2.671616	2.531630	2.379307	2.260085	2.17498	2.116034	2.074912	2.045118	2.022213	2.003407	...	1.889966	1
min	0.072700	0.327300	0.630300	1.013000	1.42450	1.698200	1.807300	1.885000	1.942100	1.988200	...	1.489600	1
25%	1.220600	1.923100	2.619300	3.076300	3.66070	4.214400	4.510300	4.711300	4.851600	4.928000	...	4.220400	4
50%	4.126300	4.501300	4.635400	4.873300	5.17140	5.496900	5.756000	5.931500	6.057000	6.148100	...	5.662900	5
75%	6.063800	6.453800	6.700200	6.920700	7.11000	7.331200	7.519800	7.634300	7.720400	7.797700	...	7.518200	7
max	9.813800	9.887800	10.145600	10.459900	10.64990	10.741400	10.766300	10.747500	10.701500	10.725100	...	11.324200	11

8 rows × 31 columns

In [4]:
```python
### There are 282 instances which is equal to 3.37% of target variable missing. I am going to drop those instances.
df_tbd.isnull().sum().sort_values(ascending=False).head()
```

Out[4]:
```
Adj_Close    282
SVENF15        0
SVENF02        0
SVENF03        0
SVENF04        0
dtype: int64
```

In [5]:
```python
print((282/df_tbd.shape[0])*100)
df_tbd.dropna(subset=["Adj_Close"],how="all",inplace=True)
```

```
3.3760325631509636
```

Prepare the Data

In [6]:
```python
### Prepare the Data
X=df_tbd.drop("Adj_Close",axis=1).values
y=df_tbd["Adj_Close"].values
X_col=df_tbd.drop("Adj_Close",axis=1).columns
### Scale the date
X=scale(X)
y=scale(y)
```

In [7]:
```python
Correlation_Mat=np.corrcoef(df_tbd[X_col].values.T)
```

SVENF19	0.822026	0.884211	0.922945	0.945085	0.955767	0.959664	0.96053	0.960859	0.961988	0.964413	0.968115	0.972804	0.97808	0.98351
SVENF20	0.822472	0.884583	0.922683	0.943506	0.952442	0.954503	0.953703	0.952695	0.952886	0.95479	0.958371	0.963308	0.969158	0.97545
SVENF21	0.821254	0.883311	0.920891	0.940525	0.947814	0.948087	0.94562	0.94323	0.942413	0.943709	0.947076	0.952168	0.95851	0.96559
SVENF22	0.818245	0.880253	0.917414	0.93599	0.941744	0.940299	0.936188	0.9324	0.930524	0.93114	0.934211	0.939374	0.946125	0.95392
SVENF23	0.81338	0.875325	0.912155	0.929804	0.934145	0.931069	0.925358	0.920175	0.917211	0.917093	0.919796	0.92495	0.93203	0.94046
SVENF24	0.80665	0.868506	0.90508	0.921925	0.92498	0.920375	0.913126	0.906572	0.902507	0.901615	0.903891	0.908965	0.916294	0.92528
SVENF25	0.7981	0.85983	0.896211	0.912369	0.914267	0.908243	0.899534	0.891646	0.886484	0.884793	0.886595	0.891523	0.899029	0.90850
SVENF26	0.787826	0.849384	0.88562	0.901202	0.902072	0.894745	0.884664	0.875496	0.869255	0.866751	0.868042	0.87277	0.880383	0.89025
SVENF27	0.775963	0.8373	0.87343	0.888536	0.888503	0.879994	0.868636	0.858251	0.850962	0.847646	0.848398	0.852879	0.860537	0.87074
SVENF28	0.762679	0.823744	0.859797	0.874519	0.873704	0.864134	0.851601	0.840071	0.831773	0.827655	0.827852	0.832046	0.839691	0.85015
SVENF29	0.748167	0.80891	0.844907	0.859327	0.857846	0.847334	0.83373	0.821133	0.811874	0.806972	0.806606	0.810482	0.818064	0.82871
SVENF30	0.732632	0.793006	0.828962	0.843153	0.841114	0.829776	0.815205	0.801622	0.791457	0.785796	0.784866	0.7884	0.795876	0.80666

Linear Regression Model

Model 1.1: Without Eleminating Correlated Features

In [9]:
```
### SPllit the data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.15,random_state=42)
X_features=df_tbd[X_col].columns
```

In [10]:
```
import time
start_time = time.time()

### chose model
linear_reg=linear_model.LinearRegression()
### fit the model
linear_reg.fit(X_train,y_train)
### predict
y_pred=linear_reg.predict(X_test)
y_pred_train=linear_reg.predict(X_train)
length_of_modeling=(time.time() - start_time)
```

In [11]:
```
### R^2 of train and test
r_square=[metrics.explained_variance_score(y_train,y_pred_train),metrics.explained_variance_score(y_test,y_pred)]
### MAE
MAE=[metrics.mean_absolute_error(y_train,y_pred_train),metrics.mean_absolute_error(y_test,y_pred)]
### MSE
MSE=[metrics.mean_squared_error(y_train,y_pred_train),metrics.mean_squared_error(y_test,y_pred)]
## RMSE
RMSE=[np.sqrt(metrics.mean_squared_error(y_train,y_pred_train)),np.sqrt(metrics.mean_squared_error(y_test,y_pred))]
```

In [12]:
```
## Measure Accuracy Molde 1.1 (Using All Features)
df_lr_acc1=pd.DataFrame({
    "Index":["Train","Test"],
"R^2":r_square,"MAE":MAE,"MSE":MSE,
"RMSE": RMSE})
df_lr_acc1["Model"]="Linear Regression All Features"
df_lr_acc1["Computation (s)"]=length_of_modeling
df_lr_acc1
```

Out[12]:

	Index	R^2	MAE	MSE	RMSE	Model	Computation (s)
0	Train	0.902273	0.249414	0.097213	0.311789	Linear Regression All Features	0.009986
1	Test	0.904134	0.251227	0.098649	0.314084	Linear Regression All Features	0.009986

In [13]:
```
### Coeficient of Linear Regression
pd.DataFrame(linear_reg.coef_,X_features).T
```

Out[13]:

	SVENF01	SVENF02	SVENF03	SVENF04	SVENF05	SVENF06	SVENF07	SVENF08	SVENF09	SVENF10	...	SVENF21	SVENF22	SVENF
0	-5.143278	53.3088	-234.759846	526.530805	-589.532938	190.492611	236.945233	-244.275422	-35.352577	253.215819	...	408.409194	-280.292783	127.1504

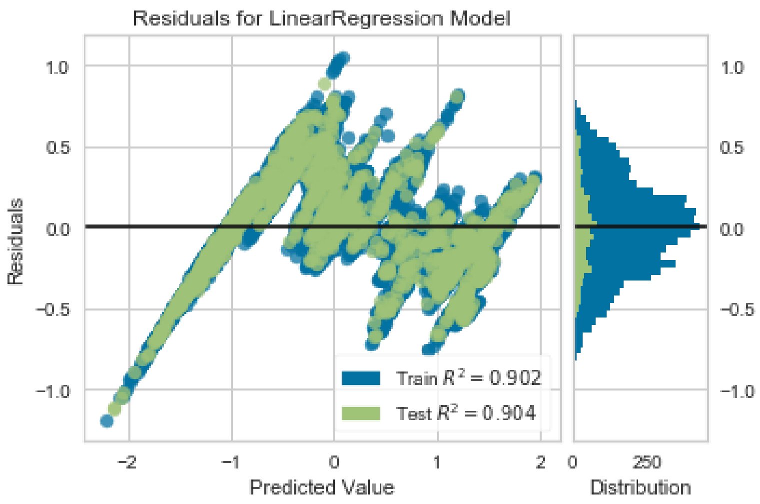
1 rows × 30 columns

In [14]:
```
pd.DataFrame({"Intercept":[linear_reg.intercept_]})
```

Out[14]:

	Intercept
0	-0.000254

In [15]:
```
visualizer=ResidualsPlot(linear_reg)
visualizer.fit(X_train, y_train)  # Fit the training data to the visualizer
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
visualizer.poof()
```



Out[15]: `<matplotlib.axes._subplots.AxesSubplot at 0x1f2b5397c18>`

Linear Regression Model

Model 1.2: after dropping correlated features

In [16]:
```
del X, X_train, X_test, y, y_train , y_test, y_pred, y_pred_train
```

In [17]:
```
df_corr_matrix=df_tbd[X_col].corr().abs()
```

In [18]:
```
# Select upper triangle of correlation matrix
upper=df_corr_matrix.where(np.triu(np.ones(df_corr_matrix.shape), k=1).astype(np.bool))
# Find index of feature columns with correlation greater than 0.80
to_drop = [column for column in upper.columns if any(upper[column] > 0.75)]
```

In [19]:
```
X=df_tbd[X_col].drop(to_drop,axis=1).values
y=df_tbd["Adj_Close"].values
X_features=df_tbd[X_col].drop(to_drop,axis=1).columns
### Scale the date
X=scale(X)
y=scale(y)
```

In [20]:
```
### SPLlit the data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.15,random_state=42)
```

In [21]:
```
### chose model
linear_reg=linear_model.LinearRegression()
### fit the model
linear_reg.fit(X_train,y_train)
### predict
y_pred=linear_reg.predict(X_test)
y_pred_train=linear_reg.predict(X_train)
length_of_modeling=(time.time() - start_time)
```

In [22]:
```
### R^2 of train and test
r_square=[metrics.explained_variance_score(y_train,y_pred_train),metrics.explained_variance_score(y_test,y_pred)]
### MAE
MAE=[metrics.mean_absolute_error(y_train,y_pred_train),metrics.mean_absolute_error(y_test,y_pred)]
### MSE
MSE=[metrics.mean_squared_error(y_train,y_pred_train),metrics.mean_squared_error(y_test,y_pred)]
## RMSE
RMSE=[np.sqrt(metrics.mean_squared_error(y_train,y_pred_train)),np.sqrt(metrics.mean_squared_error(y_test,y_pred))]
```

In [23]:
```
pd.DataFrame(linear_reg.coef_,X_features).T
```

Out[23]:

	SVENF01
0	-0.847919

In [24]:
```
pd.DataFrame({"Intercept":[linear_reg.intercept_]})
```

Out[24]:

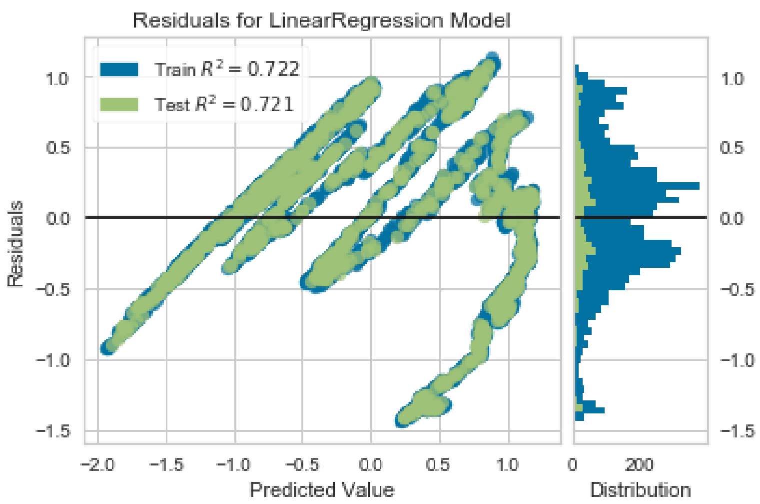
	Intercept
0	-0.002149

In [25]:
```
## Measure Accuracy Molde 1.1
df_lr_acc2=pd.DataFrame({
    "Index":["Train","Test"],
"R^2":r_square,"MAE":MAE,"MSE":MSE,
"RMSE": RMSE})
df_lr_acc2["Model"]="Linear Reg Exlcluding Correlated Features"
df_lr_acc2["Computation (s)"]=length_of_modeling
df_lr_acc2
```

Out[25]:

	Index	R^2	MAE	MSE	RMSE	Model	Computation (s)
0	Train	0.721765	0.417872	0.27677	0.526090	Linear Reg Exlcluding Correlated Features	1.032239
1	Test	0.721636	0.425444	0.28664	0.535388	Linear Reg Exlcluding Correlated Features	1.032239

In [26]:
```
visualizer=ResidualsPlot(linear_reg)
visualizer.fit(X_train, y_train)  # Fit the training data to the visualizer
visualizer.score(X_test, y_test)  # Evaluate the model on the test data
visualizer.poof()
```



Out[26]: `<matplotlib.axes._subplots.AxesSubplot at 0x1f2b5b074e0>`

SVR: Before Running PCA

1. SVR_linear
2. SV_rbf

SVR_linear

In [27]:
```
### Prepare the Data
del X, X_train, X_test, y, y_train , y_test, y_pred, y_pred_train
```

In [28]:
```
### Prepare the Data
X=df_tbd.drop("Adj_Close",axis=1).values
y=df_tbd["Adj_Close"].values
X_col=df_tbd.drop("Adj_Close",axis=1).columns
X_features=df_tbd[X_col].columns
### Scale the date
X=scale(X)
y=scale(y)
```

In [29]:
```
### SPllit the data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.15,random_state=42)
```

In [30]:
```
### Load the Model
start_time = time.time()
svr_linear = SVR(kernel='linear')
## Fit the model
svr_linear.fit(X_train,y_train)
### predict
y_pred=svr_linear.predict(X_test)
y_pred_train=svr_linear.predict(X_train)
length_of_modeling=(time.time() - start_time)
```

In [31]:
```
### R^2 of train and test
r_square=[metrics.explained_variance_score(y_train,y_pred_train),metrics.explained_variance_score(y_test,y_pred)]
### MAE
MAE=[metrics.mean_absolute_error(y_train,y_pred_train),metrics.mean_absolute_error(y_test,y_pred)]
### MSE
MSE=[metrics.mean_squared_error(y_train,y_pred_train),metrics.mean_squared_error(y_test,y_pred)]
## RMSE
RMSE=[np.sqrt(metrics.mean_squared_error(y_train,y_pred_train)),np.sqrt(metrics.mean_squared_error(y_test,y_pred))]
```

In [32]:
```
pd.DataFrame(svr_linear.coef_,columns=X_features)
```

Out[32]:

	SVENF01	SVENF02	SVENF03	SVENF04	SVENF05	SVENF06	SVENF07	SVENF08	SVENF09	SVENF10	...	SVENF21	SVENF22	SVENF23	SVENF24	SVE
0	0.31224	-0.26275	-1.89201	-0.057542	1.383026	1.50333	0.754565	-0.213348	-0.954514	-1.287506	...	-0.12549	-0.561058	-0.923559	-1.160391	-1.21

1 rows × 30 columns

In [33]:
```
pd.DataFrame({"Intercept":[svr_linear.intercept_]})
```

Out[33]:

	Intercept
0	[-0.003328551657788893]

In [34]:
```
## Measure Accuracy Molde 1.1
df_svr_lr=pd.DataFrame({
    "Index":["Train","Test"],
"R^2":r_square,"MAE":MAE,"MSE":MSE,
"RMSE": RMSE})
df_svr_lr["Model"]="SVR Linear"
df_svr_lr["Computation (s)"]=length_of_modeling
df_svr_lr
```

Out[34]:

	Index	R^2	MAE	MSE	RMSE	Model	Computation (s)
0	Train	0.893919	0.251190	0.105535	0.324861	SVR Linear	11.587043
1	Test	0.894682	0.254702	0.108379	0.329209	SVR Linear	11.587043

SVR_rfb

In [35]:
```
### Prepare the Data
del X, X_train, X_test, y, y_train , y_test, y_pred, y_pred_train
```

In [36]:
```
### Prepare the Data
X=df_tbd.drop("Adj_Close",axis=1).values
y=df_tbd["Adj_Close"].values
X_col=df_tbd.drop("Adj_Close",axis=1).columns
X_features=df_tbd[X_col].columns
### Scale the date
X=scale(X)
y=scale(y)
```

In [37]:
```
### SPllit the data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.15,random_state=42)
```

In [38]:
```
### Load the Model
start_time = time.time()
svr_rbf = SVR(kernel='rbf',gamma='scale')
## Fit the model
svr_rbf.fit(X_train,y_train)
### predict
y_pred=svr_rbf.predict(X_test)
y_pred_train=svr_rbf.predict(X_train)
length_of_modeling=(time.time() - start_time)
```

In [39]:
```
### R^2 of train and test
r_square=[metrics.explained_variance_score(y_train,y_pred_train),metrics.explained_variance_score(y_test,y_pred)]
### MAE
MAE=[metrics.mean_absolute_error(y_train,y_pred_train),metrics.mean_absolute_error(y_test,y_pred)]
### MSE
MSE=[metrics.mean_squared_error(y_train,y_pred_train),metrics.mean_squared_error(y_test,y_pred)]
## RMSE
RMSE=[np.sqrt(metrics.mean_squared_error(y_train,y_pred_train)),np.sqrt(metrics.mean_squared_error(y_test,y_pred))]
```

In [40]:
```
## Measure Accuracy Molde 1.1
df_svr_rfb=pd.DataFrame({
    "Index":["Train","Test"],
"R^2":r_square,"MAE":MAE,"MSE":MSE,
"RMSE": RMSE})
df_svr_rfb["Model"]="SVR RBF"
df_svr_rfb["Computation (s)"]=length_of_modeling
df_svr_rfb
```

Out[40]:

	Index	R^2	MAE	MSE	RMSE	Model	Computation (s)
0	Train	0.989421	0.073756	0.010539	0.102658	SVR RBF	1.157941
1	Test	0.989597	0.073983	0.010734	0.103605	SVR RBF	1.157941

PCA

Select first 3 Features that explain the data

In [41]:
```
### Prepare the Data
del X, X_train, X_test, y, y_train , y_test, y_pred, y_pred_train
X=df_tbd.drop("Adj_Close",axis=1).values
y=df_tbd["Adj_Close"].values
X_col=df_tbd.drop("Adj_Close",axis=1).columns
X_features=df_tbd[X_col].columns
### Scale the date
X=scale(X)
y=scale(y)
```

In [42]:
```
##Load the Model
pca=decomposition.PCA()
treasury_pca=pca.fit_transform(X)
```

In [43]:
```
#Change the Number Fromat of DATA frame
pd.options.display.float_format = '{:,.4f}'.format
df_exvar_ratio=pd.DataFrame(pca.explained_variance_ratio_*100,X_features).reset_index().rename(columns={
    "index":"Features",0:"explained_variance_ratio"})
### Sort
df_exvar_ratio.sort_values("explained_variance_ratio",ascending=False,inplace=True)
## Cumulative Sume
df_exvar_ratio["explained_variance_ratio_cumsum"]=df_exvar_ratio["explained_variance_ratio"].cumsum()
```

Select first 3 Features that explain the data

In [44]:
```
#### Featurs Selections
list_of_PCA_features=list(df_exvar_ratio.iloc[:3]["Features"])
print("First 3 Features Explains", df_exvar_ratio.iloc[:3]["explained_variance_ratio_cumsum"].max(),"% of the dataset")
```

```
First 3 Features Explains 99.43469070485438 % of the dataset
```

Linear Regression Model with PCA

In [45]:
```
del X, y,X_features ,X_col
```

In [46]:
```
### Prepare the Data
X=df_tbd[list_of_PCA_features].values
y=df_tbd["Adj_Close"].values
X_col=df_tbd[list_of_PCA_features].columns
X_features=df_tbd[list_of_PCA_features].columns
### Scale the date
X=scale(X)
y=scale(y)
```

In [47]:
```
### SPllit the data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.15,random_state=42)
```

In [48]:
```
### chose model
start_time = time.time()

linear_reg=linear_model.LinearRegression()
### fit the model
linear_reg.fit(X_train,y_train)
### predict
y_pred=linear_reg.predict(X_test)
y_pred_train=linear_reg.predict(X_train)
length_of_modeling=(time.time() - start_time)
```

In [49]:
```
### R^2 of train and test
r_square=[metrics.explained_variance_score(y_train,y_pred_train),metrics.explained_variance_score(y_test,y_pred)]
### MAE
MAE=[metrics.mean_absolute_error(y_train,y_pred_train),metrics.mean_absolute_error(y_test,y_pred)]
### MSE
MSE=[metrics.mean_squared_error(y_train,y_pred_train),metrics.mean_squared_error(y_test,y_pred)]
## RMSE
RMSE=[np.sqrt(metrics.mean_squared_error(y_train,y_pred_train)),np.sqrt(metrics.mean_squared_error(y_test,y_pred))]
```

In [50]:
```
## Measure Accuracy Molde 1.1 (Using All Features)
df_lr_acc_pca=pd.DataFrame({
    "Index":["Train","Test"],
"R^2":r_square,"MAE":MAE,"MSE":MSE,
"RMSE": RMSE})
df_lr_acc_pca["Model"]="Linear Regression 3 PCA Features"
df_lr_acc_pca["Computation (s)"]=length_of_modeling
df_lr_acc_pca
```

Out[50]:

	Index	R^2	MAE	MSE	RMSE	Model	Computation (s)
0	Train	0.8126	0.3681	0.1864	0.4318	Linear Regression 3 PCA Features	0.0020
1	Test	0.8111	0.3779	0.1944	0.4409	Linear Regression 3 PCA Features	0.0020

In [51]:
```
### Coeficient of Linear Regression
pd.DataFrame(linear_reg.coef_,X_features).T
```

Out[51]:

	SVENF01	SVENF02	SVENF03
0	-0.5403	1.5370	-1.9036

In [52]:
```
pd.DataFrame({"Intercept":[linear_reg.intercept_]})
```

Out[52]:

	Intercept
0	0.0003

SVR: After Running PCA

1. SVR_linear
2. SV_rbf

SVR Linear PCA

In [53]:
```
### Prepare the Data
del X, X_train, X_test, y, y_train , y_test, y_pred, y_pred_train
```

In [54]:
```
### Prepare the Data
X=df_tbd[list_of_PCA_features].values
y=df_tbd["Adj_Close"].values
X_col=df_tbd[list_of_PCA_features].columns
X_features=df_tbd[list_of_PCA_features].columns
### Scale the date
X=scale(X)
y=scale(y)
```

In [55]:
```
### SPLlit the data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.15,random_state=42)
```

In [56]:
```
### Load the Model
start_time = time.time()
svr_linear = SVR(kernel='linear')
## Fit the model
svr_linear.fit(X_train,y_train)
### predict
y_pred=svr_linear.predict(X_test)
y_pred_train=svr_linear.predict(X_train)
length_of_modeling=(time.time() - start_time)
```

In [57]:
```
### R^2 of train and test
r_square=[metrics.explained_variance_score(y_train,y_pred_train),metrics.explained_variance_score(y_test,y_pred)]
### MAE
MAE=[metrics.mean_absolute_error(y_train,y_pred_train),metrics.mean_absolute_error(y_test,y_pred)]
### MSE
MSE=[metrics.mean_squared_error(y_train,y_pred_train),metrics.mean_squared_error(y_test,y_pred)]
## RMSE
RMSE=[np.sqrt(metrics.mean_squared_error(y_train,y_pred_train)),np.sqrt(metrics.mean_squared_error(y_test,y_pred))]
```

In [58]:
```
## Measure Accuracy Molde 1.1
df_svr_lr_pca=pd.DataFrame({
    "Index":["Train","Test"],
"R^2":r_square,"MAE":MAE,"MSE":MSE,
"RMSE": RMSE})
df_svr_lr_pca["Model"]="SVR Linear 3 PCA Features"
df_svr_lr_pca["Computation (s)"]=length_of_modeling
```

SVR RBF PCA

In [59]:
```
### Prepare the Data
del X, X_train, X_test, y, y_train , y_test, y_pred, y_pred_train
```

In [60]:
```
### Prepare the Data
X=df_tbd[list_of_PCA_features].values
y=df_tbd["Adj_Close"].values
X_col=df_tbd[list_of_PCA_features].columns
X_features=df_tbd[list_of_PCA_features].columns
### Scale the date
X=scale(X)
y=scale(y)
```

In [61]:
```
### SPLlit the data
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.15,random_state=42)
```

In [62]:
```
### Load the Model
start_time = time.time()
svr_rbf = SVR(kernel='rbf',gamma='scale')
## Fit the model
svr_rbf.fit(X_train,y_train)
### predict
y_pred=svr_rbf.predict(X_test)
y_pred_train=svr_rbf.predict(X_train)
length_of_modeling=(time.time() - start_time)
```

In [63]:
```
### R^2 of train and test
r_square=[metrics.explained_variance_score(y_train,y_pred_train),metrics.explained_variance_score(y_test,y_pred)]
### MAE
MAE=[metrics.mean_absolute_error(y_train,y_pred_train),metrics.mean_absolute_error(y_test,y_pred)]
### MSE
MSE=[metrics.mean_squared_error(y_train,y_pred_train),metrics.mean_squared_error(y_test,y_pred)]
## RMSE
RMSE=[np.sqrt(metrics.mean_squared_error(y_train,y_pred_train)),np.sqrt(metrics.mean_squared_error(y_test,y_pred))]
```

In [64]:
```
## Measure Accuracy Model
df_svr_rfb_pca=pd.DataFrame({
    "Index":["Train","Test"],
"R^2":r_square,"MAE":MAE,"MSE":MSE,
"RMSE": RMSE})
df_svr_rfb_pca["Model"]="SVR RFB 3 PCA Features"
df_svr_rfb_pca["Computation (s)"]=length_of_modeling
```

Report Results

In [65]:
```
df_result=pd.concat(
    [df_lr_acc1,df_lr_acc2,df_svr_rfb,df_lr_acc_pca,df_svr_lr,df_svr_lr_pca,df_svr_rfb_pca]).sort_values(
    ["Model","Index"])
df_result
```

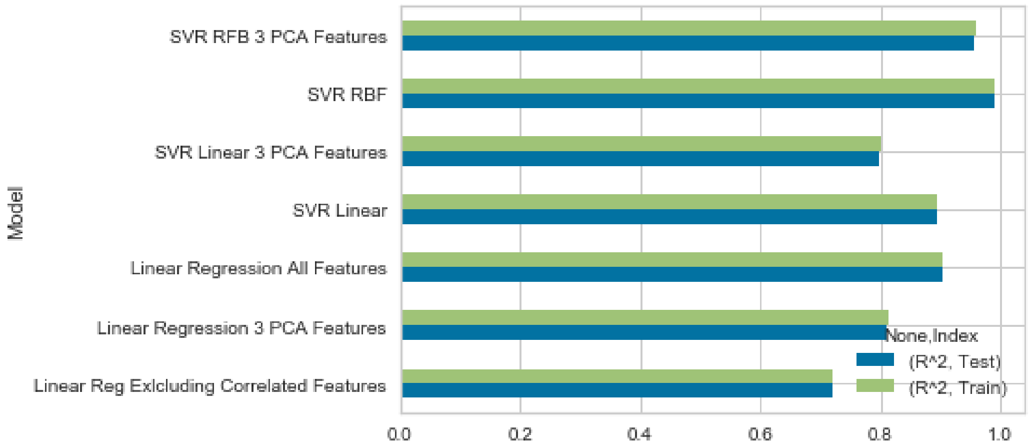
Out[65]:

	Index	R^2	MAE	MSE	RMSE	Model	Computation (s)
1	Test	0.7216	0.4254	0.2866	0.5354	Linear Reg Exlcluding Correlated Features	1.0322
0	Train	0.7218	0.4179	0.2768	0.5261	Linear Reg Exlcluding Correlated Features	1.0322
1	Test	0.8111	0.3779	0.1944	0.4409	Linear Regression 3 PCA Features	0.0020
0	Train	0.8126	0.3681	0.1864	0.4318	Linear Regression 3 PCA Features	0.0020
1	Test	0.9041	0.2512	0.0986	0.3141	Linear Regression All Features	0.0100
0	Train	0.9023	0.2494	0.0972	0.3118	Linear Regression All Features	0.0100
1	Test	0.8947	0.2547	0.1084	0.3292	SVR Linear	11.5870
0	Train	0.8939	0.2512	0.1055	0.3249	SVR Linear	11.5870
1	Test	0.7989	0.3633	0.2101	0.4584	SVR Linear 3 PCA Features	2.0027
0	Train	0.8016	0.3545	0.2003	0.4475	SVR Linear 3 PCA Features	2.0027
1	Test	0.9896	0.0740	0.0107	0.1036	SVR RBF	1.1579
0	Train	0.9894	0.0738	0.0105	0.1027	SVR RBF	1.1579
1	Test	0.9579	0.1394	0.0439	0.2096	SVR RFB 3 PCA Features	1.0894
0	Train	0.9581	0.1337	0.0421	0.2052	SVR RFB 3 PCA Features	1.0894

Residual Plots of All Models

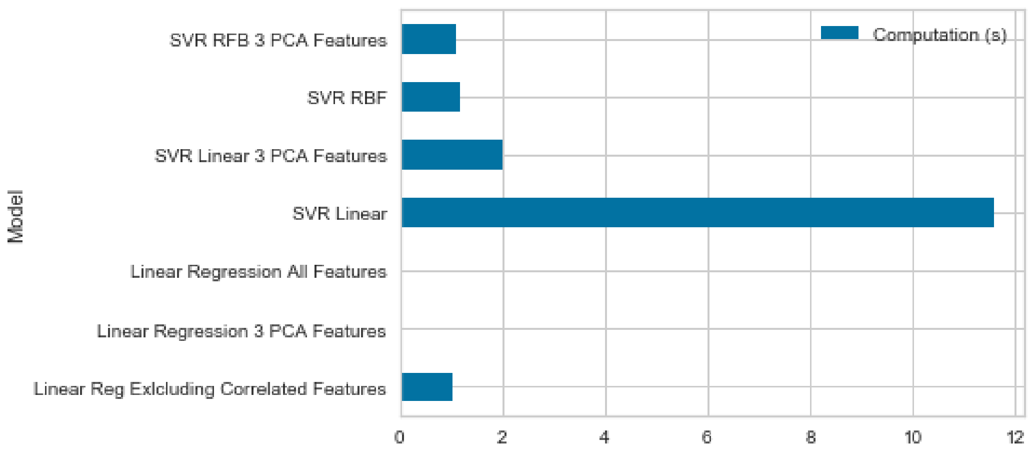
In [66]:
```python
#Reshape Melt or decast Dataframe by using pivot_table
df_result.pivot_table(
    index=['Model'],
    columns=['Index'],values=['R^2']).plot.barh()
```

Out[66]: `<matplotlib.axes._subplots.AxesSubplot at 0x1f2b55cf4a8>`



In [67]:
```python
df_result.set_index("Model").drop(
    ["Index","R^2","MAE","MSE","RMSE"],axis=1).drop_duplicates().plot.barh()
```

Out[67]: `<matplotlib.axes._subplots.AxesSubplot at 0x1f2b5c4f898>`



In [87]:
```python
he SVR linear fuction computation length from 11 seconds is \nreduced to 2 seconds while its accuracy from 89% is reduced to 80%")
```

```
The SVR rbf model performance SVR rbf is the best among all the models and it about 10 times less expensive than
SVR Linear Model. After applying PCA and selecting 3 features the SVR linear fuction computation length from 11 seconds is
reduced to 2 seconds while its accuracy from 89% is reduced to 80%
```

In [69]:
```python
print("My name is Farbod Baharkoush")
print("My NetID is: fbahar2")
print("I hereby certify that I have read the University policy on Academic Integrity and that I am not in violation.")
```

```
My name is Farbod Baharkoush
My NetID is: fbahar2
I hereby certify that I have read the University policy on Academic Integrity and that I am not in violation.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: