In [1]:
```
#importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
#import statsmodels.api as sm

import warnings
warnings.filterwarnings('ignore')
```

Reading the Data

In [2]:
```
data = pd.read_csv('Flood_claims_data.csv')
data.head()# year month date were extracted
```

Out[2]:

	incidentType	declarationDate	disasterNumber	county	damagedStateAbbreviation	damagedCit
0	Flood	2008-05-09T00:00:00.000Z	1755	Aroostook (County)	ME	FORT KEN
1	Flood	2008-05-09T00:00:00.000Z	1755	Aroostook (County)	ME	FORT KEN
2	Flood	2008-05-09T00:00:00.000Z	1755	Aroostook (County)	ME	EAGLE LAK
3	Flood	2008-05-09T00:00:00.000Z	1755	Aroostook (County)	ME	VAN BUREI
4	Flood	2008-05-09T00:00:00.000Z	1755	Aroostook (County)	ME	ISLAND FALL

5 rows × 71 columns

In [3]:
```
df=data.copy() #making a deep copy of df
df
```

Out[3]:

	incidentType	declarationDate	disasterNumber	county	damagedStateAbbreviation
0	Flood	2008-05-09T00:00:00.000Z	1755	Aroostook (County)	ME
1	Flood	2008-05-09T00:00:00.000Z	1755	Aroostook (County)	ME
2	Flood	2008-05-09T00:00:00.000Z	1755	Aroostook (County)	ME
3	Flood	2008-05-09T00:00:00.000Z	1755	Aroostook (County)	ME
4	Flood	2008-05-09T00:00:00.000Z	1755	Aroostook (County)	ME
...	...	...	...	...	...
846251	Flood	2004-04-21T00:00:00.000Z	1512	Essex (County) (in PMSA 1120,4160,7090)	MA
846252	Flood	2004-04-21T00:00:00.000Z	1512	Essex (County) (in PMSA 1120,4160,7090)	MA
846253	Flood	2004-04-21T00:00:00.000Z	1512	Essex (County) (in PMSA 1120,4160,7090)	MA
846254	Flood	2004-04-21T00:00:00.000Z	1512	Suffolk (County)	MA
846255	Flood	2004-04-21T00:00:00.000Z	1512	Middlesex (County)(in (P)MSA 1120,2600,4560)	MA

846256 rows × 71 columns

Data Preprocessing

In [4]:
```
df.shape    # gives shape of the pandas dataframe
```

Out[4]: (846256, 71)

In [5]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 846256 entries, 0 to 846255
Data columns (total 71 columns):
 #   Column                          Non-Null Count   Dtype  
---  ------                          --------------   -----  
 0   incidentType                    846256 non-null  object 
 1   declarationDate                 846256 non-null  object 
 2   disasterNumber                  846256 non-null  int64  
 3   county                          846256 non-null  object 
 4   damagedStateAbbreviation        846256 non-null  object 
 5   damagedCity                     846256 non-null  object 
 6   damagedZipCode                  846256 non-null  int64  
 7   applicantAge                    846194 non-null  object 
 8   householdComposition            846256 non-null  object 
 9   occupantsUnderTwo               846256 non-null  object 
 10  occupants2to5                   846256 non-null  object 
 11  occupants6to18                  846256 non-null  object 
 12  occupants19to64                 846256 non-null  object 
 13  occupants65andOver              846256 non-null  object 
 14  grossIncome                     846255 non-null  object 
 15  ownRent                         846256 non-null  object 
 16  primaryResidence                840630 non-null  float64
 17  residenceType                   846256 non-null  object 
 18  homeOwnersInsurance             846256 non-null  int64  
 19  floodInsurance                  846256 non-null  int64  
 20  registrationMethod              843613 non-null  object 
 21  ihpReferral                     846256 non-null  int64  
 22  ihpEligible                     846256 non-null  int64  
 23  ihpAmount                       846256 non-null  float64
 24  fipAmount                       846256 non-null  int64  
 25  haReferral                      846256 non-null  int64  
 26  haEligible                      846256 non-null  int64  
 27  haAmount                        846256 non-null  float64
 28  haStatus                        718723 non-null  object 
 29  onaReferral                     846256 non-null  int64  
 30  onaEligible                     846256 non-null  int64  
 31  onaAmount                       846256 non-null  float64
 32  utilitiesOut                    843915 non-null  float64
 33  homeDamage                      829661 non-null  float64
 34  autoDamage                      846254 non-null  float64
 35  emergencyNeeds                  846249 non-null  float64
 36  foodNeed                        190915 non-null  float64
 37  shelterNeed                     119850 non-null  float64
 38  accessFunctionalNeeds           846256 non-null  int64  
 39  sbaEligible                     0 non-null       float64
 40  sbaApproved                     846256 non-null  int64  
 41  inspnIssued                     846256 non-null  int64  
 42  inspnReturned                   846256 non-null  int64  
 43  habitabilityRepairsRequired     661930 non-null  float64
 44  rpfvl                           846256 non-null  float64
 45  ppfvl                           846256 non-null  float64
 46  renterDamageLevel               76623 non-null   object 
 47  destroyed                       846256 non-null  int64  
 48  waterLevel                      846256 non-null  int64  
 49  highWaterLocation               525910 non-null  object 
 50  floodDamage                     846256 non-null  int64  
 51  floodDamageAmount               846256 non-null  float64
 52  foundationDamage                846256 non-null  int64  
 53  foundationDamageAmount          846256 non-null  float64
 54  roofDamage                      846256 non-null  int64  
```

```
 55  roofDamageAmount               846256 non-null  float64
 56  tsaEligible                    846256 non-null  int64
 57  tsaCheckedIn                   846256 non-null  int64
 58  rentalAssistanceEligible       846256 non-null  int64
 59  rentalAssistanceAmount         846256 non-null  float64
 60  repairAssistanceEligible       846256 non-null  int64
 61  repairAmount                   846256 non-null  float64
 62  replacementAssistanceEligible  846256 non-null  int64
 63  replacementAmount              846256 non-null  float64
 64  personalPropertyEligible       846256 non-null  int64
 65  personalPropertyAmount         846256 non-null  float64
 66  ihpMax                         846256 non-null  int64
 67  haMax                          846256 non-null  int64
 68  onaMax                         846256 non-null  int64
 69  lastRefresh                    846256 non-null  object
 70  id                             846256 non-null  object
dtypes: float64(21), int64(29), object(21)
memory usage: 458.4+ MB
None
```

In [6]:
```
df.columns[df.isnull().any()]# looking for missing values in dataframe
```

Out[6]:
```
Index(['applicantAge', 'grossIncome', 'primaryResidence', 'registrationMethod',
       'haStatus', 'utilitiesOut', 'homeDamage', 'autoDamage',
       'emergencyNeeds', 'foodNeed', 'shelterNeed', 'sbaEligible',
       'habitabilityRepairsRequired', 'renterDamageLevel',
       'highWaterLocation'],
      dtype='object')
```

In [7]:
```
df.loc[:,['applicantAge', 'grossIncome', 'primaryResidence', 'registrationMethod',
       'haStatus', 'utilitiesOut', 'homeDamage', 'autoDamage',
       'emergencyNeeds', 'foodNeed', 'shelterNeed', 'sbaEligible',
       'habitabilityRepairsRequired', 'renterDamageLevel',
       'highWaterLocation']].isna().sum()# since there are too many  missing values ir
```

Out[7]:
```
applicantAge                     62
grossIncome                       1
primaryResidence               5626
registrationMethod             2643
haStatus                     127533
utilitiesOut                   2341
homeDamage                    16595
autoDamage                        2
emergencyNeeds                    7
foodNeed                     655341
shelterNeed                  726406
sbaEligible                  846256
habitabilityRepairsRequired  184326
renterDamageLevel            769633
highWaterLocation            320346
dtype: int64
```

In [8]:
```
#drop unnessisary col
df.drop(['habitabilityRepairsRequired', 'haStatus', 'renterDamageLevel',
       'highWaterLocation','declarationDate','lastRefresh','sbaEligible','shelterNeed'
df.shape
```

Out[8]:
```
(846256, 62)
```

In [9]:
```
df.columns[df.isnull().any()]# looking for missing values in dataframe
```

Out[9]:
```
Index(['applicantAge', 'grossIncome', 'primaryResidence', 'registrationMethod',
       'utilitiesOut', 'homeDamage', 'autoDamage', 'emergencyNeeds'],
      dtype='object')
```

In [10]:
```
df.loc[:,['applicantAge', 'grossIncome', 'primaryResidence', 'registrationMethod',
       'utilitiesOut', 'homeDamage', 'autoDamage', 'emergencyNeeds']].isna().sum()
```

Out[10]:
```
applicantAge            62
grossIncome              1
primaryResidence      5626
registrationMethod    2643
utilitiesOut          2341
homeDamage           16595
autoDamage               2
emergencyNeeds           7
dtype: int64
```

In [11]:
```
#Replace the missing values with median for
miscol=['primaryResidence','utilitiesOut', 'homeDamage', 'autoDamage', 'emergencyNeeds

for col in miscol:
    df[col].fillna(df[col].median(), inplace=True)

df.dropna(inplace=True) #for thes rest of object type missing values
```

In [12]:
```
df.loc[:,['applicantAge', 'grossIncome', 'primaryResidence', 'registrationMethod',
       'utilitiesOut', 'homeDamage', 'autoDamage', 'emergencyNeeds']].isna().sum()
```

Out[12]:
```
applicantAge          0
grossIncome           0
primaryResidence      0
registrationMethod    0
utilitiesOut          0
homeDamage            0
autoDamage            0
emergencyNeeds        0
dtype: int64
```

In [13]:
```
# one hot encoder

df = pd.get_dummies(df, columns = ['applicantAge','grossIncome','ownRent','residenceTy
df.iloc[:,58:]=df.iloc[:,58:].astype(int)
df=df.drop(['registrationMethod_Mobile','residenceType_Travel Trailer','ownRent_Unknow
df
```

Out[13]:

	incidentType	disasterNumber	county	damagedStateAbbreviation	damagedCity	damaged
0	Flood	1755	Aroostook (County)	ME	FORT KENT	
1	Flood	1755	Aroostook (County)	ME	FORT KENT	
2	Flood	1755	Aroostook (County)	ME	EAGLE LAKE	
3	Flood	1755	Aroostook (County)	ME	VAN BUREN	
4	Flood	1755	Aroostook (County)	ME	ISLAND FALLS	
...	...	...	...	...	...	
844282	Flood	4709	Broward (County)	FL	FORT LAUDERDALE	
844283	Flood	4709	Broward (County)	FL	NORTH LAUDERDALE	
844284	Flood	4709	Broward (County)	FL	FORT LAUDERDALE	
844285	Flood	4709	Broward (County)	FL	FORT LAUDERDALE	
844286	Flood	4709	Broward (County)	FL	HOLLYWOOD	

843611 rows × 85 columns

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 843611 entries, 0 to 844286
Data columns (total 85 columns):
 #   Column                                Non-Null Count   Dtype  
---  ------                                --------------   -----  
 0   incidentType                          843611 non-null  object 
 1   disasterNumber                        843611 non-null  int64  
 2   county                                843611 non-null  object 
 3   damagedStateAbbreviation              843611 non-null  object 
 4   damagedCity                           843611 non-null  object 
 5   damagedZipCode                        843611 non-null  int64  
 6   householdComposition                  843611 non-null  object 
 7   occupantsUnderTwo                     843611 non-null  object 
 8   occupants2to5                         843611 non-null  object 
 9   occupants6to18                        843611 non-null  object 
 10  occupants19to64                       843611 non-null  object 
 11  occupants65andOver                    843611 non-null  object 
 12  primaryResidence                      843611 non-null  float64
 13  homeOwnersInsurance                   843611 non-null  int64  
 14  floodInsurance                        843611 non-null  int64  
 15  ihpReferral                           843611 non-null  int64  
 16  ihpEligible                           843611 non-null  int64  
 17  ihpAmount                             843611 non-null  float64
 18  fipAmount                             843611 non-null  int64  
 19  haReferral                            843611 non-null  int64  
 20  haEligible                            843611 non-null  int64  
 21  haAmount                              843611 non-null  float64
 22  onaReferral                           843611 non-null  int64  
 23  onaEligible                           843611 non-null  int64  
 24  onaAmount                             843611 non-null  float64
 25  utilitiesOut                          843611 non-null  float64
 26  homeDamage                            843611 non-null  float64
 27  autoDamage                            843611 non-null  float64
 28  emergencyNeeds                        843611 non-null  float64
 29  accessFunctionalNeeds                 843611 non-null  int64  
 30  sbaApproved                           843611 non-null  int64  
 31  inspnIssued                           843611 non-null  int64  
 32  inspnReturned                         843611 non-null  int64  
 33  rpfvl                                 843611 non-null  float64
 34  ppfvl                                 843611 non-null  float64
 35  destroyed                             843611 non-null  int64  
 36  waterLevel                            843611 non-null  int64  
 37  floodDamage                           843611 non-null  int64  
 38  floodDamageAmount                     843611 non-null  float64
 39  foundationDamage                      843611 non-null  int64  
 40  foundationDamageAmount                843611 non-null  float64
 41  roofDamage                            843611 non-null  int64  
 42  roofDamageAmount                      843611 non-null  float64
 43  tsaEligible                           843611 non-null  int64  
 44  tsaCheckedIn                          843611 non-null  int64  
 45  rentalAssistanceEligible              843611 non-null  int64  
 46  rentalAssistanceAmount                843611 non-null  float64
 47  repairAssistanceEligible              843611 non-null  int64  
 48  repairAmount                          843611 non-null  float64
 49  replacementAssistanceEligible         843611 non-null  int64  
 50  replacementAmount                     843611 non-null  float64
 51  personalPropertyEligible              843611 non-null  int64  
 52  personalPropertyAmount                843611 non-null  float64
 53  ihpMax                                843611 non-null  int64  
 54  haMax                                 843611 non-null  int64  
```

```
 55  onaMax                                    843611 non-null  int64
 56  id                                        843611 non-null  object
 57  applicantAge_19-34                        843611 non-null  uint8
 58  applicantAge_35-49                        843611 non-null  int32
 59  applicantAge_50-64                        843611 non-null  int32
 60  applicantAge_65+                          843611 non-null  int32
 61  grossIncome_$120,001-$175,000             843611 non-null  int32
 62  grossIncome_$15,000-$30,000               843611 non-null  int32
 63  grossIncome_$30,001-$60,000               843611 non-null  int32
 64  grossIncome_$60,001-$120,000              843611 non-null  int32
 65  grossIncome_0                             843611 non-null  int32
 66  grossIncome_<$15,000                      843611 non-null  int32
 67  grossIncome_>$175,000                     843611 non-null  int32
 68  ownRent_Owner                             843611 non-null  int32
 69  ownRent_Renter                            843611 non-null  int32
 70  residenceType_Apartment                   843611 non-null  int32
 71  residenceType_Assisted Living Facility    843611 non-null  int32
 72  residenceType_Boat                        843611 non-null  int32
 73  residenceType_College Dorm                843611 non-null  int32
 74  residenceType_Condo                       843611 non-null  int32
 75  residenceType_Correctional Facility       843611 non-null  int32
 76  residenceType_House/Duplex                843611 non-null  int32
 77  residenceType_Military Housing            843611 non-null  int32
 78  residenceType_Mobile Home                 843611 non-null  int32
 79  residenceType_Other                       843611 non-null  int32
 80  residenceType_Townhouse                   843611 non-null  int32
 81  residenceType_Unknown                     843611 non-null  int32
 82  registrationMethod_Call Center            843611 non-null  int32
 83  registrationMethod_DSAT                   843611 non-null  int32
 84  registrationMethod_Internet               843611 non-null  int32
dtypes: float64(17), int32(27), int64(29), object(11), uint8(1)
memory usage: 461.0+ MB
```

Feature Selection using Correlation plot

In [15]:
```
dele=pd.DataFrame(df.drop(['incidentType','damagedStateAbbreviation','householdComposi
dele=dele[dele["floodDamageAmount"]>=0.1]
dele
```

Out[15]:

	floodDamageAmount
damagedZipCode	0.104457
primaryResidence	0.105705
homeOwnersInsurance	0.188682
floodInsurance	0.277723
ihpReferral	0.133564
ihpEligible	0.337573
ihpAmount	0.675841
fipAmount	0.124581
haReferral	0.158861
haEligible	0.382004
haAmount	0.669794
onaEligible	0.135125
onaAmount	0.270158
utilitiesOut	0.147345
homeDamage	0.128099
emergencyNeeds	0.126121
sbaApproved	0.179501
inspnIssued	0.165039
inspnReturned	0.165039
rpfvl	0.975857
ppfvl	0.649065
destroyed	0.349002
waterLevel	0.477629
floodDamage	0.356413
floodDamageAmount	1.000000
foundationDamageAmount	0.105938
tsaEligible	0.231915
rentalAssistanceEligible	0.476778
rentalAssistanceAmount	0.341655
repairAssistanceEligible	0.244295
repairAmount	0.589546
replacementAssistanceEligible	0.283046
replacementAmount	0.310360
personalPropertyEligible	0.119128

	floodDamageAmount
personalPropertyAmount	0.300195
ihpMax	0.413878
haMax	0.227912
ownRent_Owner	0.196890

In [16]:
```
dele.index
```

Out[16]:
```
Index(['damagedZipCode', 'primaryResidence', 'homeOwnersInsurance',
       'floodInsurance', 'ihpReferral', 'ihpEligible', 'ihpAmount',
       'fipAmount', 'haReferral', 'haEligible', 'haAmount', 'onaEligible',
       'onaAmount', 'utilitiesOut', 'homeDamage', 'emergencyNeeds',
       'sbaApproved', 'inspnIssued', 'inspnReturned', 'rpfvl', 'ppfvl',
       'destroyed', 'waterLevel', 'floodDamage', 'floodDamageAmount',
       'foundationDamageAmount', 'tsaEligible', 'rentalAssistanceEligible',
       'rentalAssistanceAmount', 'repairAssistanceEligible', 'repairAmount',
       'replacementAssistanceEligible', 'replacementAmount',
       'personalPropertyEligible', 'personalPropertyAmount', 'ihpMax', 'haMax',
       'ownRent_Owner'],
      dtype='object')
```

Final Data

In [17]:
```
# dropping all the insignificant columns
df=df.loc[:,['floodInsurance', 'damagedZipCode', 'primaryResidence', 'homeOwnersInsura
        'ihpReferral', 'ihpEligible', 'ihpAmount',
       'fipAmount', 'haReferral', 'haEligible', 'haAmount', 'onaEligible',
       'onaAmount', 'utilitiesOut', 'homeDamage', 'emergencyNeeds',
       'sbaApproved', 'inspnIssued', 'inspnReturned', 'rpfvl', 'ppfvl',
       'destroyed', 'waterLevel', 'floodDamage', 'floodDamageAmount',
       'foundationDamageAmount', 'tsaEligible', 'rentalAssistanceEligible',
       'rentalAssistanceAmount', 'repairAssistanceEligible', 'repairAmount',
       'replacementAssistanceEligible', 'replacementAmount',
       'personalPropertyEligible', 'personalPropertyAmount', 'ihpMax', 'haMax',
       'ownRent_Owner']]
df
```

Out[17]:

	floodInsurance	damagedZipCode	primaryResidence	homeOwnersInsurance	ihpReferral	ihpI
0	0	4743	1.0	0	1	
1	0	4743	1.0	0	1	
2	0	4739	1.0	1	1	
3	0	4785	1.0	1	1	
4	1	4747	1.0	1	1	
...	...	...	...	...	...	
844282	0	33315	1.0	1	1	
844283	0	33068	1.0	0	1	
844284	0	33317	1.0	1	1	
844285	0	33315	1.0	0	1	
844286	0	33024	1.0	1	1	

843611 rows × 38 columns

Data Spliting

In [18]:

```
#lets split data to test train to build model

X=df.drop(['floodDamageAmount'],axis=1)
y=df['floodDamageAmount']

#lets scale the data but its a rf model so scaling isnt nessisary
from sklearn.preprocessing import StandardScaler
Xscaler=StandardScaler()
yscaler=StandardScaler()

# X=Xscaler.fit_transform(X)
# y=yscaler.fit_transform(y.array.reshape(-1, 1))

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

Xscaler.fit(X_train)
X_train_smaller=Xscaler.transform(X_train.head(100000))
X_train=Xscaler.transform(X_train)
X_test=Xscaler.transform(X_test)

yscaler.fit(y_train.array.reshape(-1, 1))
y_train_smaller=yscaler.transform(y_train.head(100000).array.reshape(-1, 1))
y_train=yscaler.fit_transform(y_train.array.reshape(-1, 1))
y_test=yscaler.transform(y_test.array.reshape(-1, 1))

X_train
```

Out[18]:
```
array([[-0.31007334, -1.95960877,  0.27886119, ..., -0.09908103,
        -0.03981197,  0.68428723],
       [-0.31007334, -0.47683527,  0.27886119, ..., -0.09908103,
        -0.03981197,  0.68428723],
       [-0.31007334, -0.47688625,  0.27886119, ..., -0.09908103,
        -0.03981197, -1.46137463],
       ...,
       [ 3.22504348,  0.15571892,  0.27886119, ..., -0.09908103,
        -0.03981197, -1.46137463],
       [-0.31007334,  0.6700101 ,  0.27886119, ..., -0.09908103,
        -0.03981197, -1.46137463],
       [-0.31007334,  0.15526014,  0.27886119, ..., -0.09908103,
        -0.03981197,  0.68428723]])
```

Attempt to do Gridsearch

In [19]:
```
# model=RandomForestRegressor()
# model.fit(X_train,y_train)



# train_pred=model.predict(X_train)

# #lets see the models rmse

# rmse=sqrt(mean_squared_error(y_train, train_pred, squared=False))
# print(f"the root mean squared error for train data: {rmse}")
# print(f"the R2 score(Explained variance) for test data: {r2_score(y_train, train_pre
```

In [20]:
```
# test_pred=model.predict(X_test)

# #lets see the models rmse

# rmse=sqrt(mean_squared_error(y_test, test_pred, squared=False))
# print(f"the root mean squared error for test data: {rmse}")
# print(f"the R2 score(Explained variance) for test data: {r2_score(y_test, test_pred)
```

In [21]:
```
# #lets use random search for better model
# from sklearn.model_selection import RandomizedSearchCV
# params={'max_depth': [10, 50, 70, 100, None],
#  'max_features': ['auto', 'sqrt'],
#  'min_samples_leaf': [1, 2, None],
#  'min_samples_split': [2, 5, None],
#  'n_estimators': [200,1000,  1600, 2000]}

# # Use the random grid to search for best hyperparameters
# # First create the base model to tune
# rf = RandomForestRegressor()
# # Random search of parameters, using 3 fold cross validation,
# # search across 100 different combinations, and use all available cores
# rf_random = RandomizedSearchCV(estimator = rf, param_distributions = params, n_iter
# # Fit the random search model
# rf_random.fit(X_train, y_train)
# rf_random.best_params_
```

Random Forest model and its performance

In [22]:
```
#best model
#lets import RF model
from cmath import sqrt
from sklearn.metrics import r2_score
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error

model=RandomForestRegressor()
model.fit(X_train,y_train)

best_train_pred_rf=model.predict(X_train)

#lets see the models rmse

mse=mean_squared_error(y_train, best_train_pred_rf, squared=False)
print(f"the root mean squared error for train data: {mse}")
print(f"the R2 score(Explained variance) for train data: {r2_score(y_train, best_train

best_test_pred_rf=model.predict(X_test)

#lets see the models rmse

mse1=mean_squared_error(y_test, best_test_pred_rf, squared=False)
print(f"the root mean squared error for test data: {mse1}")
print(f"the R2 score(Explained variance) for test data: {r2_score(y_test, best_test_pr

# add these error values to plot error visualizations

errordf=pd.DataFrame(columns=["Regressor","Train_error","Test_error","R2 score for tes
errordf.loc[0,"Regressor"]="Random Forest"
errordf.loc[0,"Train_error"]=mse
errordf.loc[0,"Test_error"]=mse1
errordf.loc[0,"R2 score for test data"]=r2_score(y_test, best_test_pred_rf)
```

```
the root mean squared error for train data: 0.0251353795855566
the R2 score(Explained variance) for train data: 0.99936821269309
the root mean squared error for test data: 0.04934924962174929
the R2 score(Explained variance) for test data: 0.9975706030462753
```

In [23]:
```
#lets see the predicted data after scaling it

dele=pd.DataFrame(yscaler.inverse_transform(best_test_pred_rf.reshape(-1, 1)))
print(dele.shape)
dele[dele[0]>1]
```

```
(168723, 1)
```

Out[23]:

	0
5	45742.5236
6	2384.5452
8	4996.1447
9	16846.1337
10	18000.1251
...	...
168714	15029.4053
168716	12557.9697
168718	10655.5419
168719	38526.6288
168722	4908.3111

87151 rows × 1 columns

In [24]:
```
#Actual values of flood claim amounts

dele=pd.DataFrame(yscaler.inverse_transform(y_test.reshape(-1, 1)))
print(dele.shape)
dele[dele[0]>1]
```

```
(168723, 1)
```

Out[24]:

	0
5	45795.96
6	2515.64
8	5074.22
9	18781.08
10	17874.94
...	...
168714	15143.83
168716	12613.34
168718	10701.15
168719	38537.60
168722	5066.55

86904 rows × 1 columns

SVR model and its performance

In [25]:
```
# lets also build an SVR model
from sklearn.svm import SVR
regressor = SVR()
regressor.fit(X_train_smaller,y_train_smaller)

best_train_pred_svr=regressor.predict(X_train_smaller)

#lets see the models rmse

mse=mean_squared_error(y_train_smaller, best_train_pred_svr, squared=False)
print(f"the root mean squared error for train data: {mse}")
print(f"the R2 score(Explained variance) for train data: {r2_score(y_train_smaller, be

best_test_pred_svr=regressor.predict(X_test)

#lets see the models rmse

mse1=mean_squared_error(y_test, best_test_pred_svr, squared=False)
print(f"the root mean squared error for test data: {mse1}")
print(f"the R2 score(Explained variance) for test data: {r2_score(y_test, best_test_pr

# add these error values to plot error visualizations


errordf.loc[1,"Regressor"]="SVR"
errordf.loc[1,"Train_error"]=mse
errordf.loc[1,"Test_error"]=mse1
errordf.loc[1,"R2 score for test data"]=r2_score(y_test, best_test_pred_svr)
```

```
the root mean squared error for train data: 0.38908002718593915
the R2 score(Explained variance) for train data: 0.8551073457505213
the root mean squared error for test data: 0.3689418099588025
the R2 score(Explained variance) for test data: 0.8642145850589197
```

XGBoost model and its performance

In [26]:
```
#lets build a Xgboost model

import xgboost as xg
xgb_r = xg.XGBRegressor(objective ='reg:linear', seed = 123)

# Fitting the model
xgb_r.fit(X_train, y_train)


best_train_pred_xgb=xgb_r.predict(X_train)

#lets see the models rmse

mse=mean_squared_error(y_train, best_train_pred_xgb, squared=False)
print(f"the root mean squared error for train data: {mse}")
print(f"the R2 score(Explained variance) for train data: {r2_score(y_train, best_trair

best_test_pred_xgb=xgb_r.predict(X_test)

#lets see the models rmse

mse1=mean_squared_error(y_test, best_test_pred_xgb, squared=False)
```

```
print(f"the root mean squared error for test data: {mse1}")
print(f"the R2 score(Explained variance) for test data: {r2_score(y_test, best_test_pr

# add these error values to plot error visualizations


errordf.loc[2,"Regressor"]="Xgboost"
errordf.loc[2,"Train_error"]=mse
errordf.loc[2,"Test_error"]=mse1
errordf.loc[2,"R2 score for test data"]=r2_score(y_test, best_test_pred_xgb)
```

```
[05:54:53] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group
-i-07593ffd91cd9da33-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:21
3: reg:linear is now deprecated in favor of reg:squarederror.
the root mean squared error for train data: 0.033707539987651906
the R2 score(Explained variance) for train data: 0.9988638017479808
the root mean squared error for test data: 0.04800430632695914
the R2 score(Explained variance) for test data: 0.9977012180806651
```

Deep Learning model and its performance

In [27]:
```
# Lets build a deep learning model on the data

import pandas as pd
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
import seaborn as sns
from matplotlib import pyplot as plt
```

Deep Learning model with 2 hidden layers (128,64) neurons respectively

In [28]:
```
# define the model
#Experiment with deeper and wider networks
model = Sequential()
model.add(Dense(128, input_dim=37, activation='relu'))
model.add(Dense(64, activation='relu'))
#Output layer
model.add(Dense(1, activation='linear'))

model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mae'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               4864

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 13,185
Trainable params: 13,185
Non-trainable params: 0
_________________________________________________________________
```

In [29]:
```
history = model.fit(X_train, y_train, epochs =25, validation_data=(X_test, y_test))
```

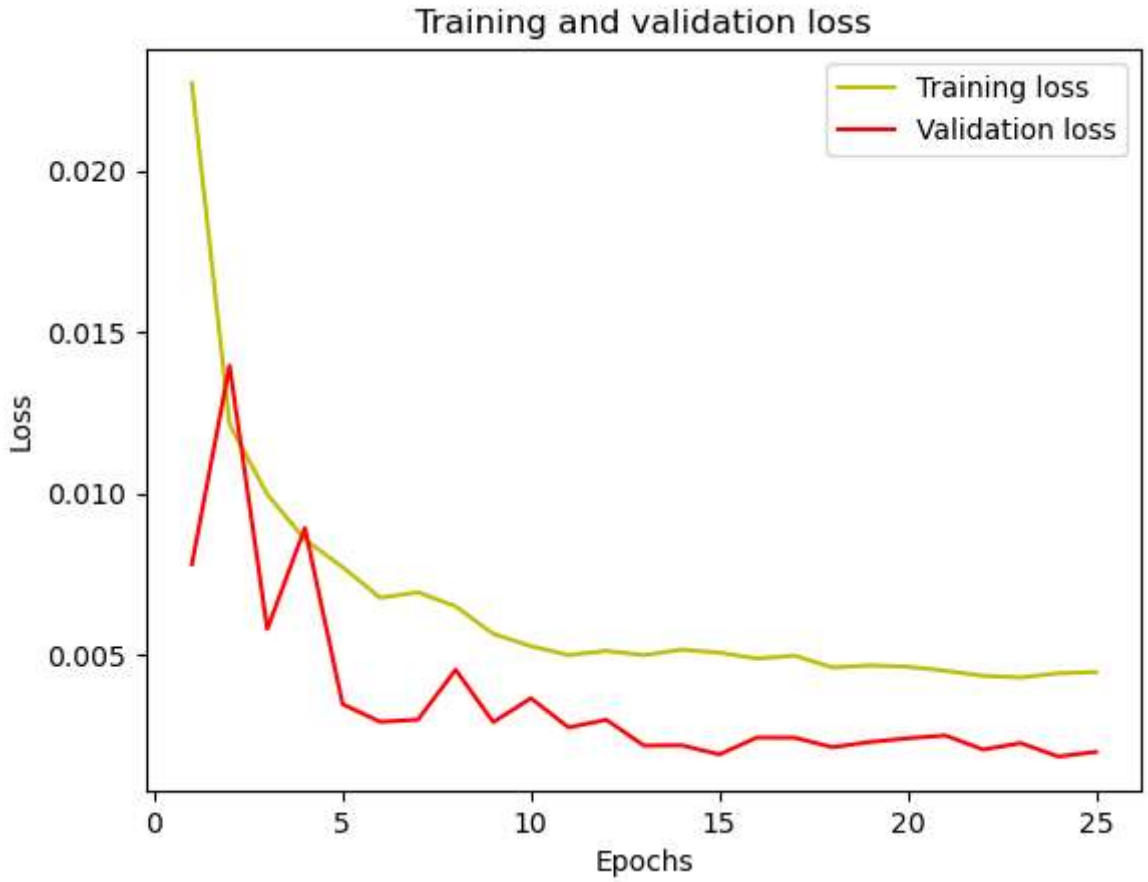
```
Epoch 1/25
21091/21091 [==============================] - 58s 3ms/step - loss: 0.0227 - mae: 0.0
391 - val_loss: 0.0078 - val_mae: 0.0273
Epoch 2/25
21091/21091 [==============================] - 58s 3ms/step - loss: 0.0121 - mae: 0.0
254 - val_loss: 0.0140 - val_mae: 0.0391
Epoch 3/25
21091/21091 [==============================] - 58s 3ms/step - loss: 0.0100 - mae: 0.0
224 - val_loss: 0.0058 - val_mae: 0.0186
Epoch 4/25
21091/21091 [==============================] - 55s 3ms/step - loss: 0.0086 - mae: 0.0
194 - val_loss: 0.0089 - val_mae: 0.0319
Epoch 5/25
21091/21091 [==============================] - 55s 3ms/step - loss: 0.0077 - mae: 0.0
178 - val_loss: 0.0035 - val_mae: 0.0131
Epoch 6/25
21091/21091 [==============================] - 54s 3ms/step - loss: 0.0068 - mae: 0.0
176 - val_loss: 0.0029 - val_mae: 0.0119
Epoch 7/25
21091/21091 [==============================] - 58s 3ms/step - loss: 0.0069 - mae: 0.0
176 - val_loss: 0.0030 - val_mae: 0.0123
Epoch 8/25
21091/21091 [==============================] - 55s 3ms/step - loss: 0.0065 - mae: 0.0
164 - val_loss: 0.0045 - val_mae: 0.0177
Epoch 9/25
21091/21091 [==============================] - 58s 3ms/step - loss: 0.0057 - mae: 0.0
160 - val_loss: 0.0029 - val_mae: 0.0125
Epoch 10/25
21091/21091 [==============================] - 58s 3ms/step - loss: 0.0053 - mae: 0.0
154 - val_loss: 0.0037 - val_mae: 0.0113
Epoch 11/25
21091/21091 [==============================] - 58s 3ms/step - loss: 0.0050 - mae: 0.0
150 - val_loss: 0.0028 - val_mae: 0.0126
Epoch 12/25
21091/21091 [==============================] - 55s 3ms/step - loss: 0.0051 - mae: 0.0
152 - val_loss: 0.0030 - val_mae: 0.0152
Epoch 13/25
21091/21091 [==============================] - 55s 3ms/step - loss: 0.0050 - mae: 0.0
146 - val_loss: 0.0022 - val_mae: 0.0110
Epoch 14/25
21091/21091 [==============================] - 55s 3ms/step - loss: 0.0052 - mae: 0.0
150 - val_loss: 0.0022 - val_mae: 0.0130
Epoch 15/25
21091/21091 [==============================] - 57s 3ms/step - loss: 0.0051 - mae: 0.0
146 - val_loss: 0.0019 - val_mae: 0.0106
Epoch 16/25
21091/21091 [==============================] - 54s 3ms/step - loss: 0.0049 - mae: 0.0
147 - val_loss: 0.0025 - val_mae: 0.0135
Epoch 17/25
21091/21091 [==============================] - 54s 3ms/step - loss: 0.0050 - mae: 0.0
142 - val_loss: 0.0024 - val_mae: 0.0136
Epoch 18/25
21091/21091 [==============================] - 54s 3ms/step - loss: 0.0046 - mae: 0.0
146 - val_loss: 0.0022 - val_mae: 0.0124
Epoch 19/25
21091/21091 [==============================] - 55s 3ms/step - loss: 0.0047 - mae: 0.0
142 - val_loss: 0.0023 - val_mae: 0.0156
Epoch 20/25
21091/21091 [==============================] - 58s 3ms/step - loss: 0.0046 - mae: 0.0
150 - val_loss: 0.0024 - val_mae: 0.0150
```

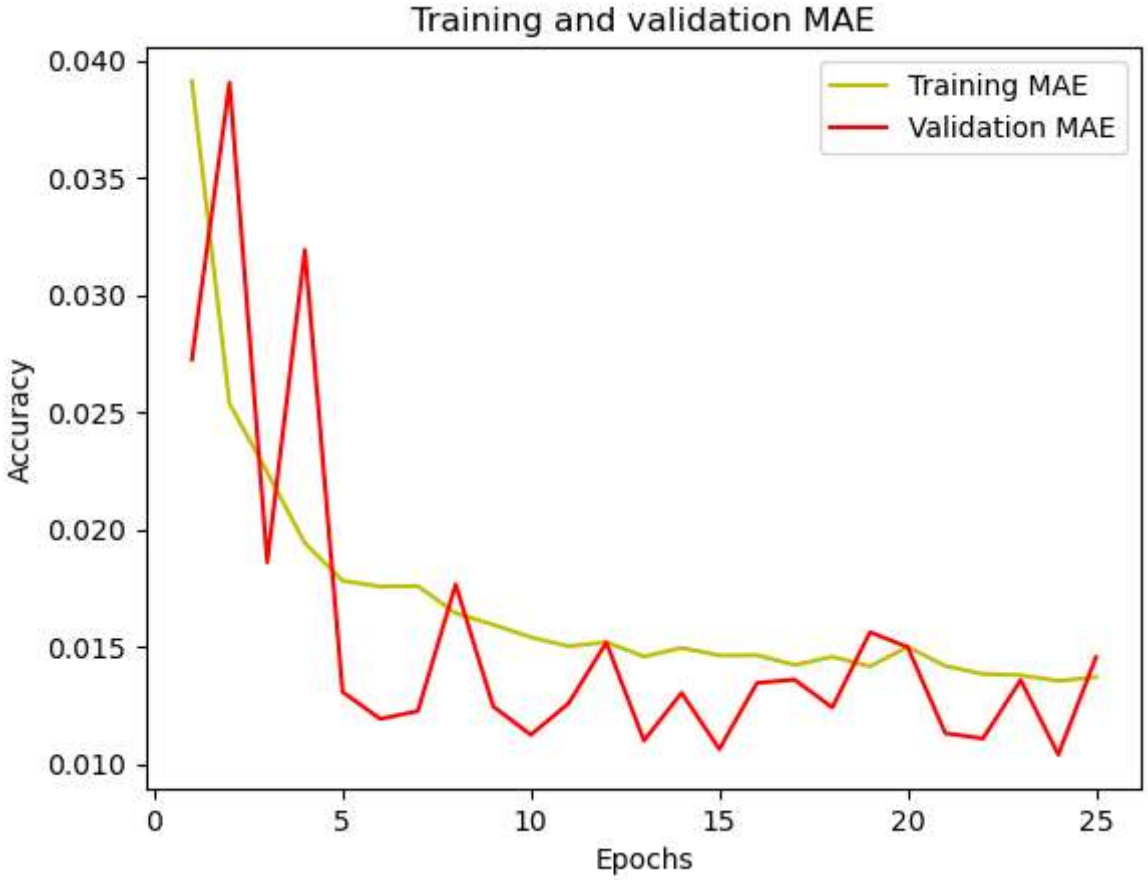
```
Epoch 21/25
21091/21091 [==============================] - 57s 3ms/step - loss: 0.0045 - mae: 0.0
142 - val_loss: 0.0025 - val_mae: 0.0113
Epoch 22/25
21091/21091 [==============================] - 58s 3ms/step - loss: 0.0044 - mae: 0.0
138 - val_loss: 0.0021 - val_mae: 0.0111
Epoch 23/25
21091/21091 [==============================] - 58s 3ms/step - loss: 0.0043 - mae: 0.0
138 - val_loss: 0.0023 - val_mae: 0.0136
Epoch 24/25
21091/21091 [==============================] - 58s 3ms/step - loss: 0.0044 - mae: 0.0
136 - val_loss: 0.0019 - val_mae: 0.0104
Epoch 25/25
21091/21091 [==============================] - 54s 3ms/step - loss: 0.0045 - mae: 0.0
137 - val_loss: 0.0020 - val_mae: 0.0146
```

In [30]:

```
from matplotlib import pyplot as plt
#plot the training and validation accuracy and loss at each epoch
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)
plt.plot(epochs, loss, 'y', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()


acc = history.history['mae']
val_acc = history.history['val_mae']
plt.plot(epochs, acc, 'y', label='Training MAE')
plt.plot(epochs, val_acc, 'r', label='Validation MAE')
plt.title('Training and validation MAE')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```





In [31]:
```
#Predict on test data
predictions = model.predict(X_test)
```

```
print("Predicted values are: ", predictions)
print("Real values are: ", y_test)
```

```
5273/5273 [==============================] - 7s 1ms/step
Predicted values are:  [[-0.3824616 ]
 [-0.3824616 ]
 [-0.3824616 ]
 ...
 [-0.3824616 ]
 [-0.3824616 ]
 [ 0.15922081]]
Real values are:  [[-0.38121745]
 [-0.38121745]
 [-0.38121745]
 ...
 [-0.38121745]
 [-0.38121745]
 [ 0.18850388]]
```

In [32]:
```
#Neural network - from the current code
mse_neural, mae_neural = model.evaluate(X_test, y_test)
print('Mean squared error from neural net: ', mse_neural)
print('Mean absolute error from neural net: ', mae_neural)
```

```
5273/5273 [==============================] - 8s 2ms/step - loss: 0.0020 - mae: 0.0146
Mean squared error from neural net:  0.0020054206252098083
Mean absolute error from neural net:  0.014568534679710865
```

In [33]:
```
mse_neural, mae_neural = model.evaluate(X_train, y_train)
errordf.loc[3,"Regressor"]="Deep Learning"
errordf.loc[3,"Train_error"]=mse_neural
mse_neural, mae_neural = model.evaluate(X_test, y_test)
errordf.loc[3,"Test_error"]=mse_neural
errordf.loc[3,"R2 score for test data"]=r2_score(y_test, predictions)
```

```
21091/21091 [==============================] - 34s 2ms/step - loss: 0.0037 - mae: 0.0
147
5273/5273 [==============================] - 8s 2ms/step - loss: 0.0020 - mae: 0.0146
```

Result of all the models

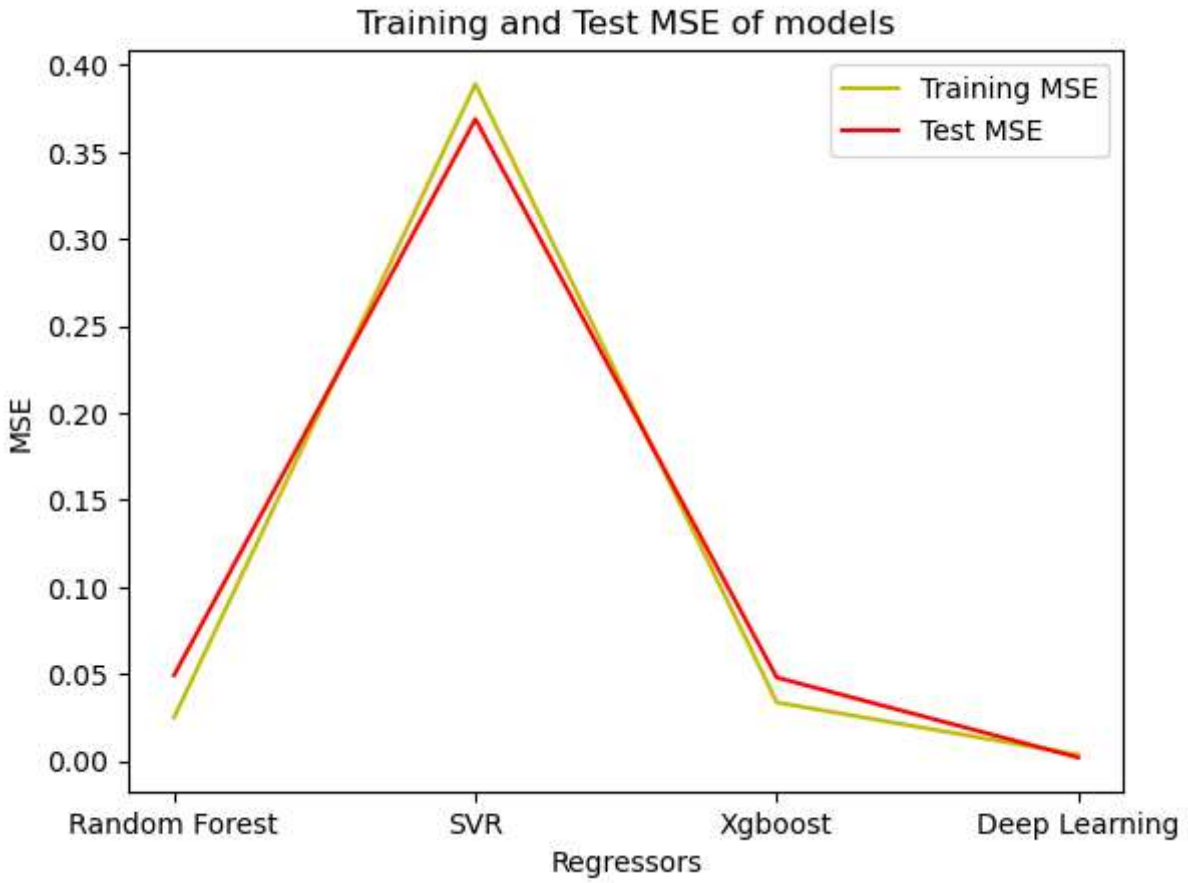
In [34]:
```
errordf
```

Out[34]:

	Regressor	Train_error	Test_error	R2 score for test data
0	Random Forest	0.025135	0.049349	0.997571
1	SVR	0.38908	0.368942	0.864215
2	Xgboost	0.033708	0.048004	0.997701
3	Deep Learning	0.003651	0.002005	0.997999

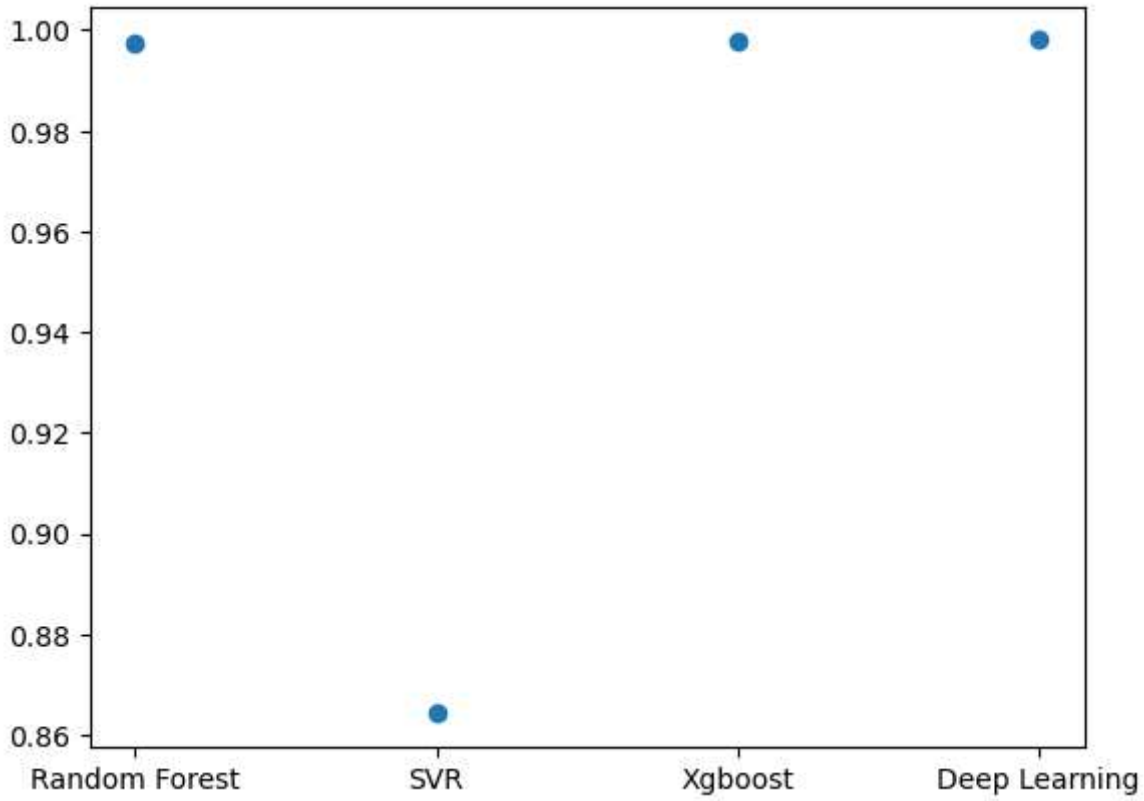
In [35]:
```
plt.plot(errordf.Regressor, errordf.Train_error, 'y', label='Training MSE')
plt.plot(errordf.Regressor, errordf.Test_error, 'r', label='Test MSE')
plt.title('Training and Test MSE of models')
plt.xlabel('Regressors')
plt.ylabel('MSE')
```

```
plt.legend()
plt.show()
```



In [36]:
```
plt.scatter(errordf.Regressor,errordf["R2 score for test data"])
```

Out[36]:
```
<matplotlib.collections.PathCollection at 0x1c999549a50>
```


1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
Random Forest
SVR
Xgboost
Deep Learning

In [ ]: