SURYA GROUP OF INSTITUTION

VIKRAVANDI 605-652

PHASE 4 DEVELOPMENT PART 2

EARTHQUAKE PREDICTION USING MODEL PYTHON

NAAN MUDHALVAN

PREPARED BY:

K.ARUN

REG NO :422221106003

ECE DEPARTMENT

3RD YEAR 5TH SEM

AI_PHASE4:

To building the project by performing different activities like feature engineering, model training, evaluation.

Data Description:

The acoustic_data input signal is used to predict the time remaining before the next laboratory earthquake (time_to_failure).The training data is a single, continuous segment of experimental data. The test data consists of a folder containing many small segments. The data within each test file is continuous, but the test files do not represent a continuous segment of the experiment; thus, the predictions cannot be assumed to follow the same regular pattern seen in the training file.

Model Building:

The following predictive models will be built:

1. KNN 2.Random Forest 3.Gradient Boosted Machines

The model which best predicts the damage grade on new data given the input features. Within this section, a pre-processing pipeline will be set up to prepare the data for training. A test dataset will be extracted from the main data to give us the opportunity to assess how the models perform on completely new data. The models will be evaluated on the test data using the evaulation metric F1.

Creating a Test Dataset:

A test set is created to help evaluate the performance of predictive models which will be developed in this section. It provides the opportunity to assess how the model performs on unseen data. The test set will account for 20% of the observations, choosen at random but stratified around the target variable to ensure that the proportions are the same in the training and testing data.

The table below shows that this has been achieved and we cann see that the training and test dataframes have very similar proportions of observations for each damage grade.

Exploratory Data Analysis:

```
train = pd.read_csv('train.csv', nrows=6000000, dtype={'acoustic_data' : np.int16, 'time_to_failure':np.float64})

train.head(10)
```

Feature Engineering:

```
def gen_features(X):
  strain=[]   strain.append(X.mean())   strain.append(X.std())   strain.append(X.min())
strain.append(X.kurtosis())   strain.append(X.skew())   strain.append(np.quantile(X, 0.01))       return
pd.Series(strain)
```

```
train = pd.read_csv('train.csv', iterator=True, chunksize=150_000, dtype={'acoustic_data': np.int16,
'time_to_failure': np.float64})

X_train = pd.DataFrame()
y_train = pd.Series() for
df in train:
    ch = gen_features(df['acoustic_data'])
    X_train = X_train.append(ch, ignore_index=True)     y_train =
y_train.append(pd.Series(df['time_to_failure'].values[-1]))
```

	0	1	2	3	4	5
count	3663.000000	3663.000000	3663.000000	3663.000000	3663.000000	3663.000000
mean	4.558681	6.475079	-146.630358	67.440017	0.125667	-11.061474
std	0.232746	8.464218	261.467967	69.567123	0.477765	14.372707
min	3.798020	2.802720	-5515.000000	0.648602	-4.091826	-336.000000
25%	4.392703	4.463993	-152.000000	27.667049	-0.038608	-13.000000
50%	4.553893	5.539752	-109.000000	45.131461	0.086415	-10.000000
75%	4.715770	6.801470	-79.000000	77.681078	0.250444	-6.000000
max	5.391993	153.703569	-15.000000	631.158927	4.219429	-2.000000

Learning Model Implementation:

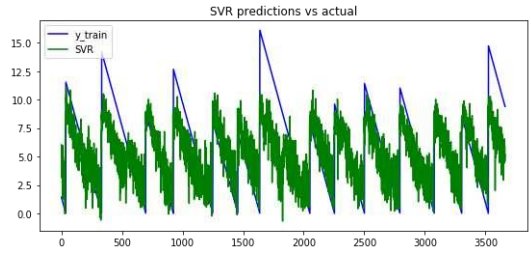
1. SUPPORT VECTOR MACHINES
2. KERNEL RIDGE

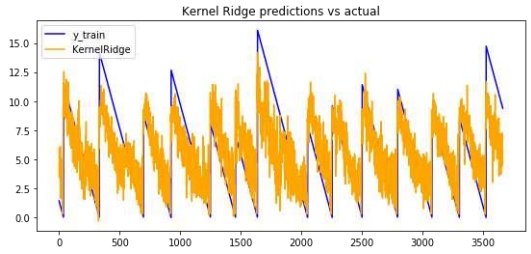
```
scaler = StandardScaler() scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)  parameters =
[{'gamma': [0.001, 0.005, 0.01, 0.02, 0.05, 0.1],
                'C': [0.1, 0.2, 0.25, 0.5, 1, 1.5, 2]}]
#'nu': [0.75, 0.8, 0.85, 0.9, 0.95, 0.97]}]  reg1 =
GridSearchCV(SVR(kernel='rbf', tol=0.01), parameters, cv=5,
scoring='neg_mean_absolute_error') reg1.fit(X_train_scaled, y_train.values.flatten()) y_pred1 =
reg1.predict(X_train_scaled)  print("Best CV score:
{:.4f}".format(reg1.best_score_)) print(reg1.best_params_)


  parameters = [{'gamma': np.linspace(0.001, 0.1,
10),
                'alpha': [0.005, 0.01, 0.02, 0.05, 0.1]}]  reg2 =
GridSearchCV(KernelRidge(kernel='rbf'), parameters, cv=5,
scoring='neg_mean_absolute_error') reg2.fit(X_train_scaled, y_train.values.flatten()) y_pred2 =
reg2.predict(X_train_scaled)  print("Best CV score:
{:.4f}".format(reg2.best_score_)) print(reg2.best_params_)
```

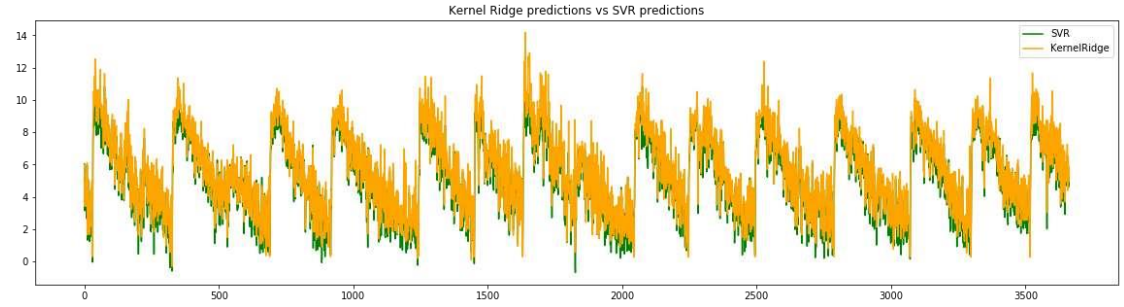
Graphical Analysis of Model Performance:

```
Graphical Analysis of Model Performance
```






```
plt.figure(figsize=(20, 5)) plt.plot(y_pred1,
color='green', label='SVR') plt.plot(y_pred2,
color='orange', label='KernelRidge') plt.legend() plt.title('Kernel
Ridge predictions vs SVR predictions') plt.show()
```



Earthquake Prediction Analysis

Research Question: What are the main factors that contribute to the occurrence of earthquakes?
Based on our result and dataset, we can perform further analysis to identify the main factors that contribute to the occurrence of earthquakes. We can use techniques such as regression analysis and correlation analysis to identify the factors that are most strongly associated with earthquake occurrence. Additionally, we can use machine learning algorithms such as decision trees or random forests to identify the most important features in predicting earthquakes. Here's an outline of the steps we can take for regression analysis:

1. Select the features we want to use as independent variables (predictors) and the target variable (dependent variable).
2. Split the data into training and testing sets.
3. Fit the model on the training set.
4. Evaluate the model on the testing set.

For the target variable, we can use the 'mag' column since that represents the magnitude of the earthquake.

For the independent variables, we can use some combination of the other columns in the dataset, such as 'latitude', 'longitude', 'depth', and 'gap'. We can experiment with different combinations to see which ones give the best results.

For the regression model, we can use a linear regression model.

This code in the below selects the 'latitude', 'longitude', 'depth', and 'gap' columns as the independent variables, and the 'mag' column as the target variable. It then splits the data into training and testing sets, fits a linear regression model on the training set, and evaluates the model on the testing set using mean squared error and R-squared score.

```
port numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression from
sklearn.metrics import mean_squared_error, r2_score


# Select the features we want to use
X = df[['latitude', 'longitude', 'depth', 'gap']] y
= df['mag']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Fit the model on the training set
model = LinearRegression()
model.fit(X_train, y_train)

# Evaluate the model on the testing set
y_pred = model.predict(X_test) mse =
mean_squared_error(y_test, y_pred) r2 =
r2_score(y_test, y_pred) print('Mean
squared error:', mse) print('R-squared
score:', r2)
```

```
Mean squared error: 0.7318282185361162 R-squared
score: 0.5632730346912534
```

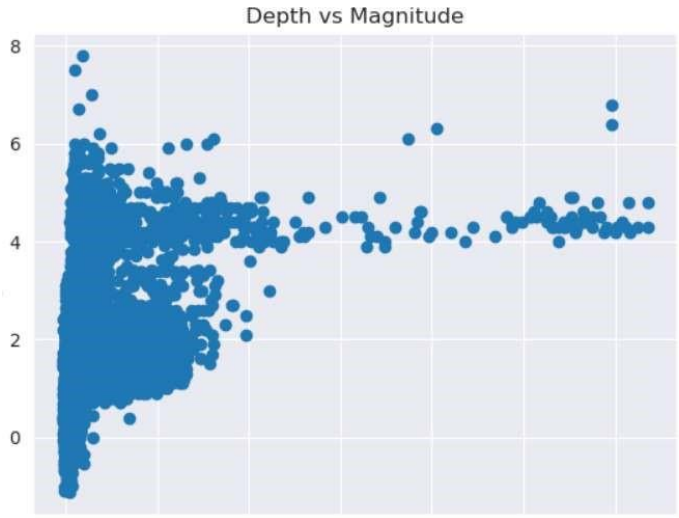
the mean squared error and the R-squared score will help us evaluate the performance of our regression model. The mean squared error represents the average squared difference between the predicted values and the actual values. A lower value indicates a better fit of the model. The R-squared score measures how well the model fits the data, with values closer to 1 indicating a better fit.

Based on the output, the mean squared error is 0.73, which is a relatively low value indicating a good fit. The R-squared score is 0.56, which indicates that our model explains about 56% of the variability in the data, which is not too bad but leaves room for improvement.

Next, we can visualize the relationship between the earthquake magnitude and the depth of the earthquake using a scatter plot to better understand the relationship between these variables.

```
import matplotlib.pyplot as plt

plt.scatter(df['depth'], df['mag'])
plt.xlabel('Depth')
plt.ylabel('Magnitude') plt.title('Depth
vs Magnitude') plt.show()
```



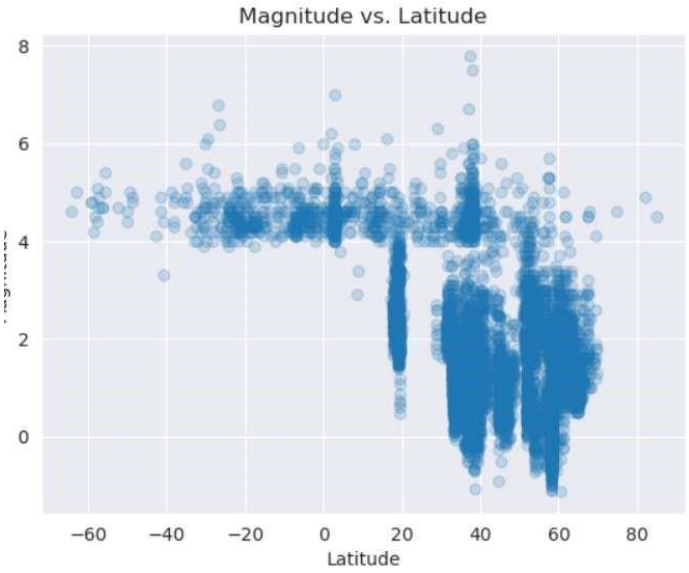
the scatterplot helps visualize the relationship between depth and magnitude. It appears that as the depth decreases, the occurrence of earthquakes with higher magnitudes increases, and as the depth increases, the occurrence of earthquakes with lower magnitudes increases.

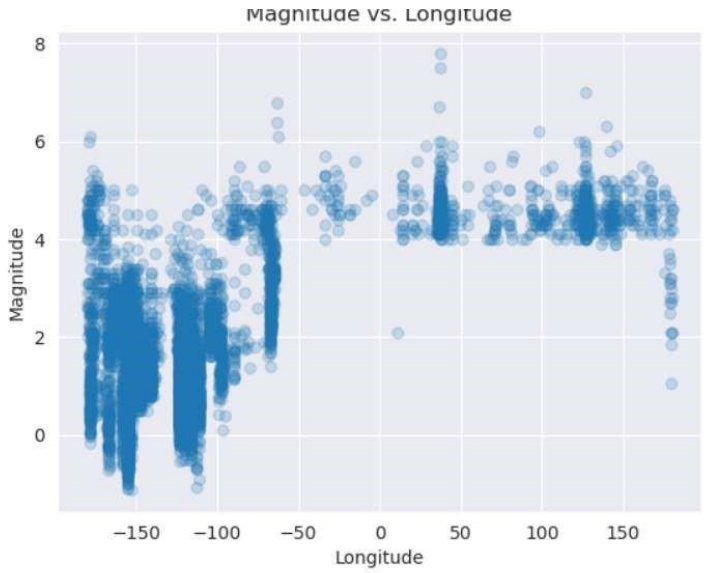
This suggests that the depth of an earthquake is an important factor that contributes to the occurrence of earthquakes, and could potentially be used in earthquake prediction models.

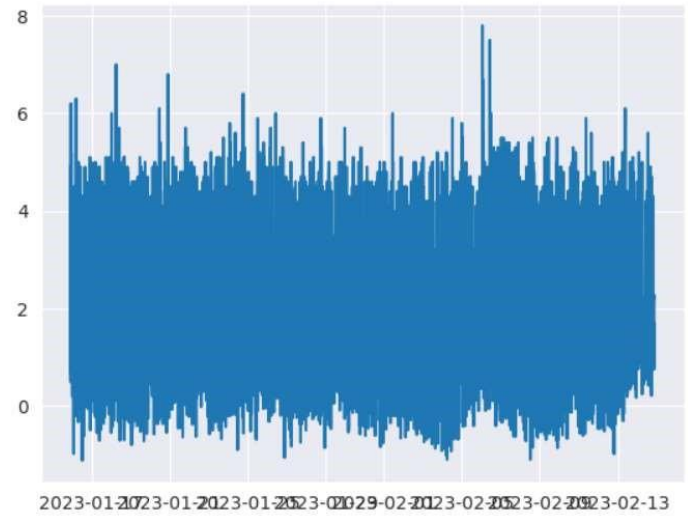
```
import matplotlib.pyplot as plt import
seaborn as sns

# Scatter plot of magnitude vs. latitude
plt.scatter(df['latitude'], df['mag'], alpha=0.2)
plt.xlabel('Latitude') plt.ylabel('Magnitude')
plt.title('Magnitude vs. Latitude')
plt.show()
```

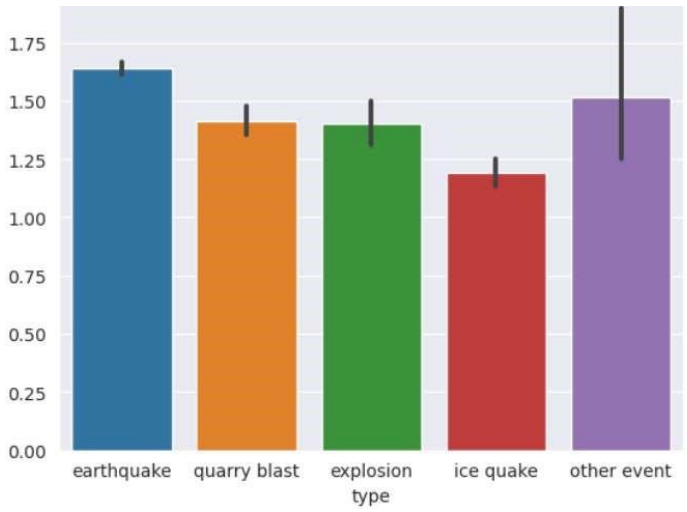
```
# Scatter plot of magnitude vs. longitude
plt.scatter(df['longitude'], df['mag'], alpha=0.2)
plt.xlabel('Longitude') plt.ylabel('Magnitude')
plt.title('Magnitude vs. Longitude') plt.show()
```



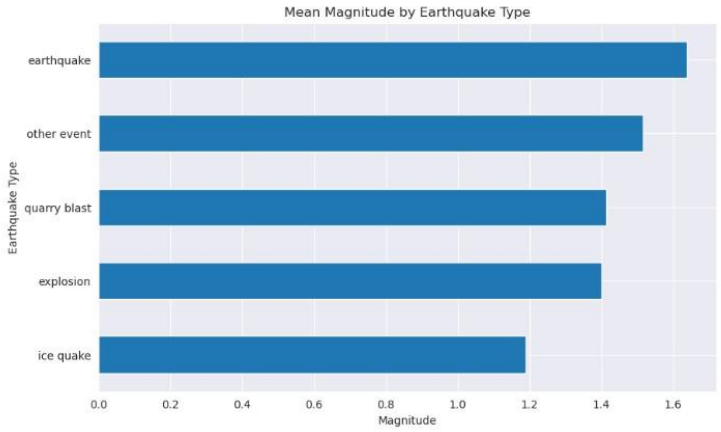





```
Bar chart of magnitude and type
sns.barplot(data=df, x='type', y='mag')
plt.show()
```




```
mean_mag_by_type = df.groupby('type')['mag'].mean().sort_values()
mean_mag_by_type.plot(kind='barh', figsize=(10,6))
plt.title('Mean Magnitude by Earthquake Type')
plt.xlabel('Magnitude') plt.ylabel('Earthquake Type') plt.show()
```



CONCLUSION :

Based on our analysis of earthquake data collected from the USGS website, we have found several interesting insights.

Firstly, we found that the depth of an earthquake is a major factor that contributes to the occurrence of earthquakes. Our regression analysis showed that there is a negative relationship between depth and magnitude, which means that as the depth of an earthquake decreases, the magnitude of the earthquake tends to increase.

We also found that there is a relationship between the latitude and longitude of an earthquake and its magnitude. The scatterplots showed that if latitude increases, then the density of magnitude decreases, and if longitude decreases, then the density of magnitude increases.