Model Development Phase Template

Date	15 April 2024
Team ID	Team - 738203
Project Title	Share Price Estimation Of TOP 5 GPU Companies
Maximum Marks	4 Marks

Initial Model Training Code, Model Validation and Evaluation Report

Initial Model Training Code:

```
# linear regression

lr=LinearRegression()
lr.fit(x_train,y_train)
lr_pred=lr.predict(x_test)

mae = mean_absolute_error(y_test,lr_pred)
mse = mean_squared_error(y_test,lr_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test,lr_pred)
print("Linear Regression Model Evaluation:")
print(f"MAE :- {mae}\n MSE :- {mse}\n RMSE:- {rmse}\n R2 Score :- {r2}")
```

```
# linear regression

lr=LinearRegression()
lr.fit(x_train,y_train)
lr_pred=lr.predict(x_test)

mae = mean_absolute_error(y_test,lr_pred)
mse = mean_squared_error(y_test,lr_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test,lr_pred)
print("Linear Regression Model Evaluation:")
print(f"MAE :- {mae}\n MSE :- {mse}\n RMSE:- {rmse}\n R2 Score :- {r2}")
```

```
# decision tree

DT=DecisionTreeRegressor()
DT.fit(x_train,y_train)
DT_pred=DT.predict(x_test)

mae = mean_absolute_error(y_test,DT_pred)
mse = mean_squared_error(y_test,DT_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test,DT_pred)
print("Decision Tree Model Evaluation:")
print(f"MAE :- {mae}\n MSE :- {mse}\n RMSE:- {rmse}\n R2 Score :- {r2}")
```

```
# Extra Trees Regression

ET=ExtraTreesRegressor()
ET.fit(x_train,y_train)
ET_pred=ET.predict(x_test)

mae = mean_absolute_error(y_test,ET_pred)
mse = mean_squared_error(y_test,ET_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test,ET_pred)
print("Extra Trees Model Evaluation:")
print(f"MAE :- {mae}\n MSE :- {mse}\n RMSE:- {rmse}\n R2 Score  :- {r2}")
```

```
#Random Forest Regression

Rf= RandomForestRegressor()
Rf.fit(x_train,y_train)
Rf_pred=Rf.predict(x_test)

mae = mean_absolute_error(y_test,Rf_pred)
mse = mean_squared_error(y_test,Rf_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test,Rf_pred)
print("Random Forest Model Evaluation:")
print(f"MAE :- {mae}\n MSE :- {mse}\n RMSE:- {rmse}\n R2 Score  :- {r2}")
```

```
# Define function to fit Prophet model
def fit_prophet(train_data):
    model = Prophet()
    train_df = train_data.rename(columns={'Date': 'ds', 'Close': 'y'})
    model.fit(train_df)
    return model

# Fit Prophet model
prophet_model = fit_prophet(train_data)

# Make predictions with Prophet
future = prophet_model.make_future_dataframe(periods=len(x_test))
# future = pd.DataFrame({'ds': pd.date_range(start=train_data.index[-1], periods=len(x_test)+1, freq='D')[1:]})
prophet_pred = prophet_model.predict(future)['yhat'].tail(len(x_test))

# Calculate evaluation metrics
mae_prophet = mean_absolute_error(y_test, prophet_pred)
mse_prophet = mean_squared_error(y_test, prophet_pred)
rmse_prophet = np.sqrt(mse_prophet)
r2_prophet = r2_score(y_test, prophet_pred)

print("Prophet Model Evaluation:")
print(f"Mean Absolute Error (MAE): {mae_prophet}")
print(f"Mean Squared Error (MSE): {mse_prophet}")
print(f"Root Mean Squared Error (RMSE): {rmse_prophet}")
print(f"R-squared (R2) Score: {r2_prophet}")
```

```python
# Arima Model

def fit_arima(train_data, order):
    model = ARIMA(train_data, order=order)
    model_fit = model.fit()
    return model_fit

arima_model = fit_arima(train_data['Close'], order=(5,1,0))
arima_pred = arima_model.predict(start=len(train_data), end=len(train_data) + len(test_data) - 1, typ='levels')

mae_arima = mean_absolute_error(test_data['Close'], arima_pred)
mse_arima = mean_squared_error(test_data['Close'], arima_pred)
rmse_arima = np.sqrt(mse_arima)
r2_arima = r2_score(test_data['Close'], arima_pred)

print("ARIMA Model Evaluation:")
print(f"Mean Absolute Error (MAE): {mae_arima}")
print(f"Mean Squared Error (MSE): {mse_arima}")
print(f"Root Mean Squared Error (RMSE): {rmse_arima}")
print(f"R-squared (R2) Score: {r2_arima}")
```

```python
#Sarimax

def fit_sarimax(train_data, order, seasonal_order):
    model = SARIMAX(train_data, order=order, seasonal_order=seasonal_order)
    model_fit = model.fit()
    return model_fit

sarimax_model = fit_sarimax(train_data['Close'], order=(1, 1, 1), seasonal_order=(1, 1, 1, 12))
sarimax_pred = sarimax_model.predict(start=len(train_data), end=len(train_data) + len(test_data) - 1, typ='levels')

mae_sarimax = mean_absolute_error(test_data['Close'], sarimax_pred)
mse_sarimax = mean_squared_error(test_data['Close'], sarimax_pred)
rmse_sarimax = np.sqrt(mse_sarimax)
r2_sarimax = r2_score(test_data['Close'], sarimax_pred)

print("SARIMAX Model Evaluation:")
print(f"Mean Absolute Error (MAE): {mae_sarimax}")
print(f"Mean Squared Error (MSE): {mse_sarimax}")
print(f"Root Mean Squared Error (RMSE): {rmse_sarimax}")
print(f"R-squared (R2) Score: {r2_sarimax}")
```

Model Validation and Evaluation Report:

Model	Accuracy/Model Evaluation

Linear Regression	Linear Regression Model Evaluation: MAE :- 0.9166458539179153 MSE :- 2.9316225864388077 RMSE:- 1.7121981738218295 R2 Score :- 0.9998367000901055
Decision Tree	Decision Tree Model Evaluation: MAE :- 4.499876777100751 MSE :- 516.4783238252011 RMSE:- 22.726159460524805 R2 Score :- 0.9712306542686407
Extra Tree	Extra Trees Model Evaluation: MAE :- 3.1554407866203813 MSE :- 541.776521284075 RMSE:- 23.276093342399086 R2 Score :- 0.9698214710454531
Random Forest	Random Forest Model Evaluation: MAE :- 4.110706619549385 MSE :- 532.7484588530283 RMSE:- 23.08134439007027 R2 Score :- 0.9703243603970128
Prophet Model	Prophet Model Evaluation: Mean Absolute Error (MAE): 195.73297116012097 Mean Squared Error (MSE): 48206.04630780599 Root Mean Squared Error (RMSE): 219.55875365788992 R-squared (R2) Score: -1.031440366400611

Arima Model	ARIMA Model Evaluation: Mean Absolute Error (MAE): 111.46898245666749 Mean Squared Error (MSE): 33882.43178759622 Root Mean Squared Error (RMSE): 184.0718114964815 R-squared (R2) Score: -0.4278320857438278
Sarimax Model	SARIMAX Model Evaluation: Mean Absolute Error (MAE): 109.07010937714938 Mean Squared Error (MSE): 32757.38967373523 Root Mean Squared Error (RMSE): 180.9900264482417 R-squared (R2) Score: -0.38042193413328573