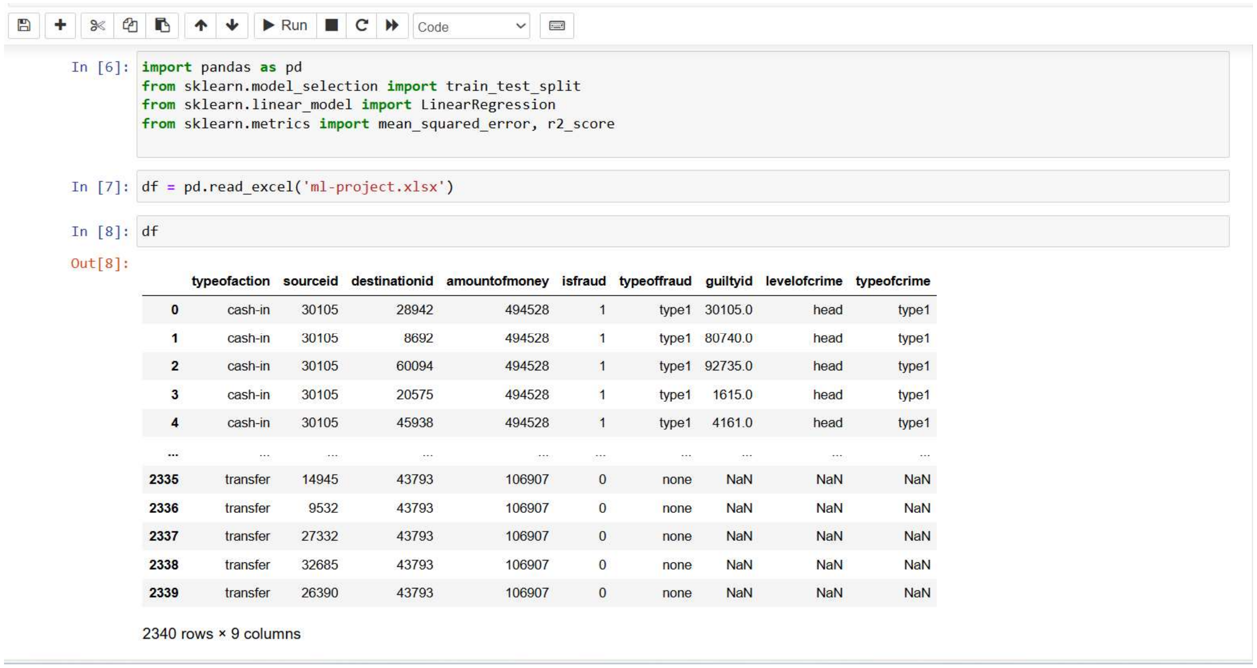


In [6]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

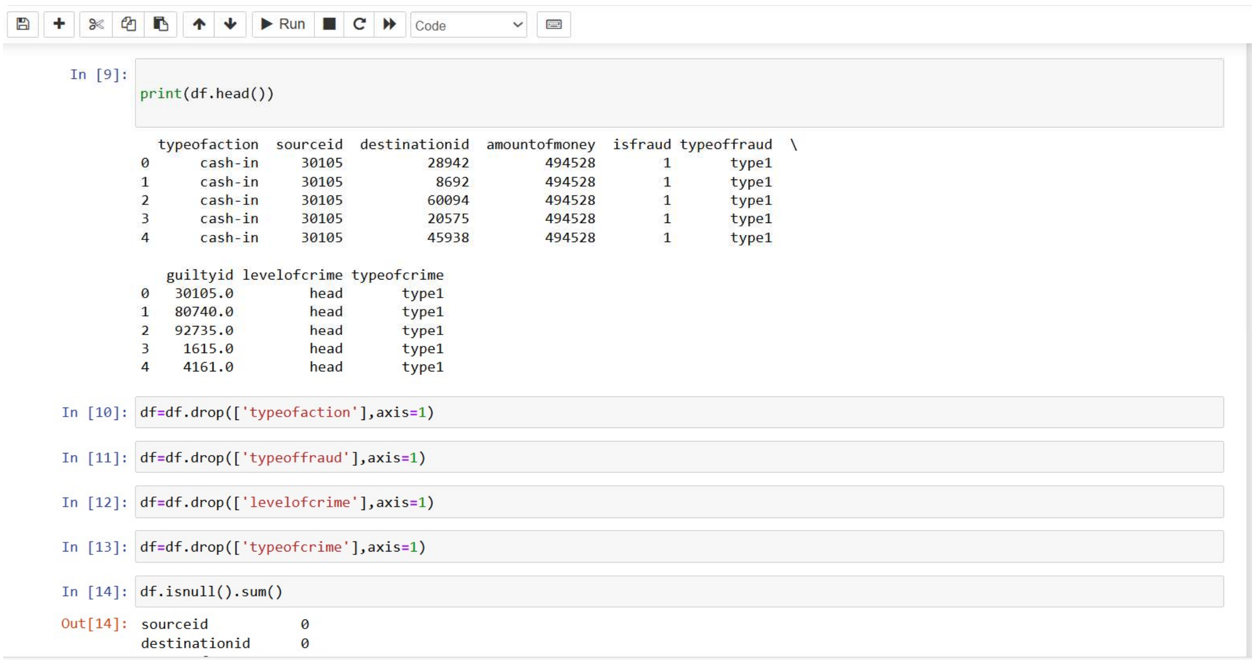
In [7]:
```python
df = pd.read_excel('ml-project.xlsx')
```

In [8]:
```python
df
```

Out[8]:

	typeofaction	sourceid	destinationid	amountofmoney	isfraud	typeoffraud	guiltyid	levelofcrime	typeofcrime
0	cash-in	30105	28942	494528	1	type1	30105.0	head	type1
1	cash-in	30105	8692	494528	1	type1	80740.0	head	type1
2	cash-in	30105	60094	494528	1	type1	92735.0	head	type1
3	cash-in	30105	20575	494528	1	type1	1615.0	head	type1
4	cash-in	30105	45938	494528	1	type1	4161.0	head	type1
...	...	...	...	...	...	...	...	...	...
2335	transfer	14945	43793	106907	0	none	NaN	NaN	NaN
2336	transfer	9532	43793	106907	0	none	NaN	NaN	NaN
2337	transfer	27332	43793	106907	0	none	NaN	NaN	NaN
2338	transfer	32685	43793	106907	0	none	NaN	NaN	NaN
2339	transfer	26390	43793	106907	0	none	NaN	NaN	NaN

2340 rows × 9 columns



In [9]:
```python
print(df.head())
```

```
  typeofaction  sourceid  destinationid  amountofmoney  isfraud typeoffraud  \
0      cash-in     30105          28942         494528        1       type1
1      cash-in     30105           8692         494528        1       type1
2      cash-in     30105          60094         494528        1       type1
3      cash-in     30105          20575         494528        1       type1
4      cash-in     30105          45938         494528        1       type1

   guiltyid levelofcrime typeofcrime
0   30105.0         head       type1
1   80740.0         head       type1
2   92735.0         head       type1
3    1615.0         head       type1
4    4161.0         head       type1
```

In [10]:
```python
df=df.drop(['typeofaction'],axis=1)
```

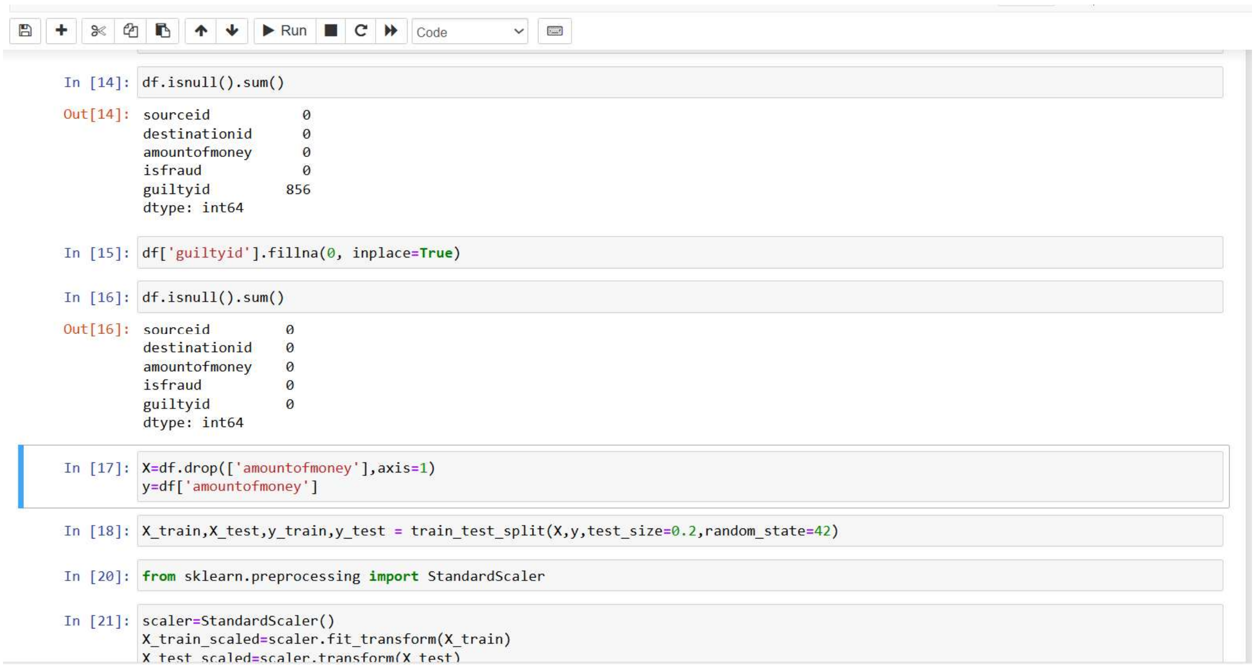
In [11]:
```python
df=df.drop(['typeoffraud'],axis=1)
```

In [12]:
```python
df=df.drop(['levelofcrime'],axis=1)
```

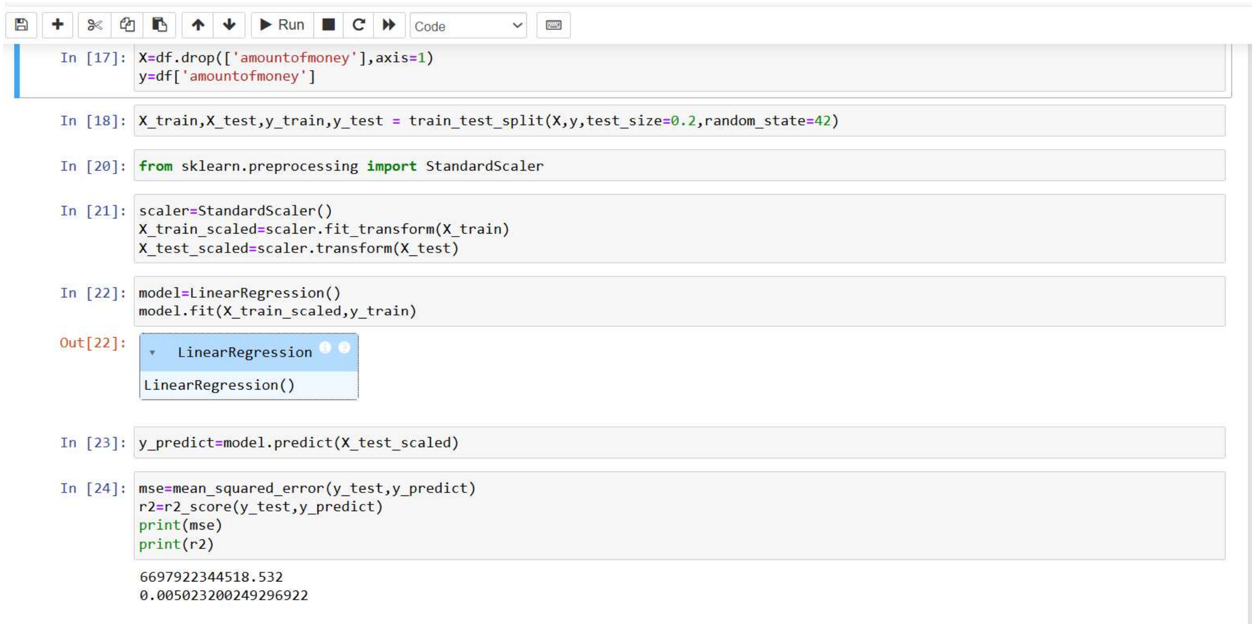
In [13]:
```python
df=df.drop(['typeofcrime'],axis=1)
```

In [14]:
```python
df.isnull().sum()
```

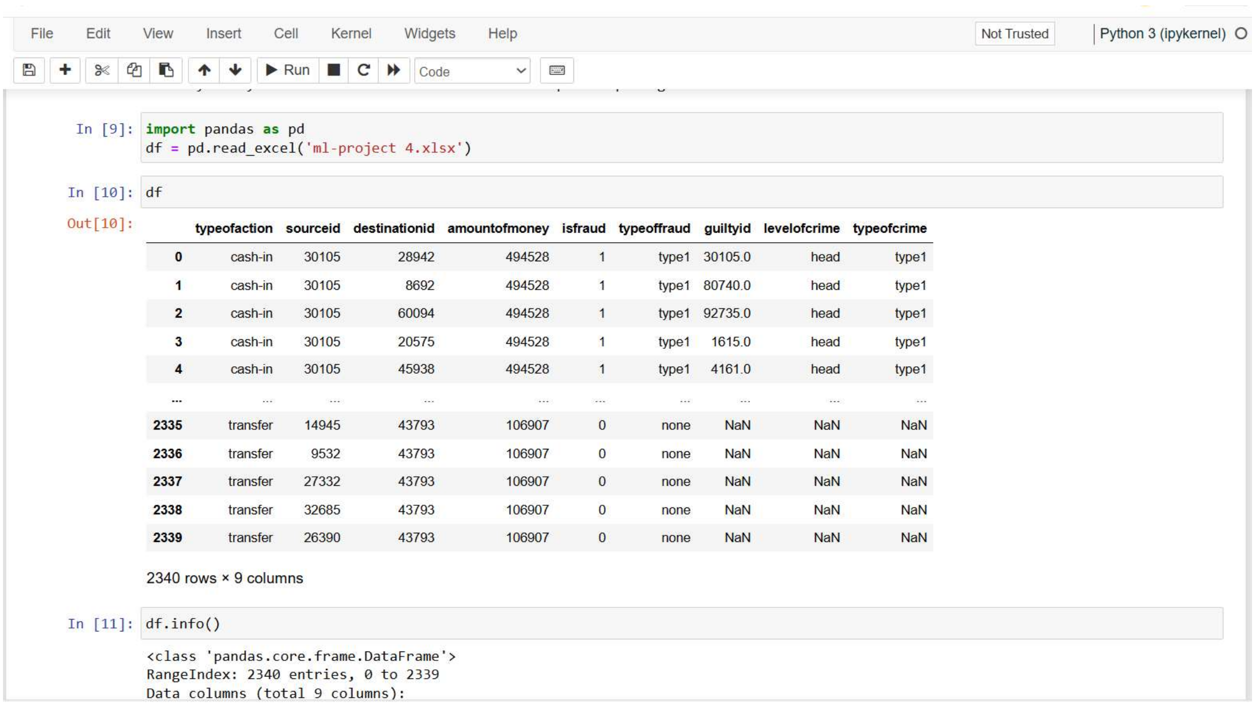
Out[14]:
```
sourceid          0
destinationid     0
```



```
In [14]: df.isnull().sum()

Out[14]: sourceid           0
         destinationid      0
         amountofmoney      0
         isfraud            0
         guiltyid         856
         dtype: int64

In [15]: df['guiltyid'].fillna(0, inplace=True)

In [16]: df.isnull().sum()

Out[16]: sourceid         0
         destinationid    0
         amountofmoney    0
         isfraud          0
         guiltyid         0
         dtype: int64

In [17]: X=df.drop(['amountofmoney'],axis=1)
         y=df['amountofmoney']

In [18]: X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=42)

In [20]: from sklearn.preprocessing import StandardScaler

In [21]: scaler=StandardScaler()
         X_train_scaled=scaler.fit_transform(X_train)
         X_test_scaled=scaler.transform(X_test)
```



```
In [17]: X=df.drop(['amountofmoney'],axis=1)
         y=df['amountofmoney']

In [18]: X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=42)

In [20]: from sklearn.preprocessing import StandardScaler

In [21]: scaler=StandardScaler()
         X_train_scaled=scaler.fit_transform(X_train)
         X_test_scaled=scaler.transform(X_test)

In [22]: model=LinearRegression()
         model.fit(X_train_scaled,y_train)

Out[22]: ▾ LinearRegression
         LinearRegression()

In [23]: y_predict=model.predict(X_test_scaled)

In [24]: mse=mean_squared_error(y_test,y_predict)
         r2=r2_score(y_test,y_predict)
         print(mse)
         print(r2)

         6697922344518.532
         0.005023200249296922
```

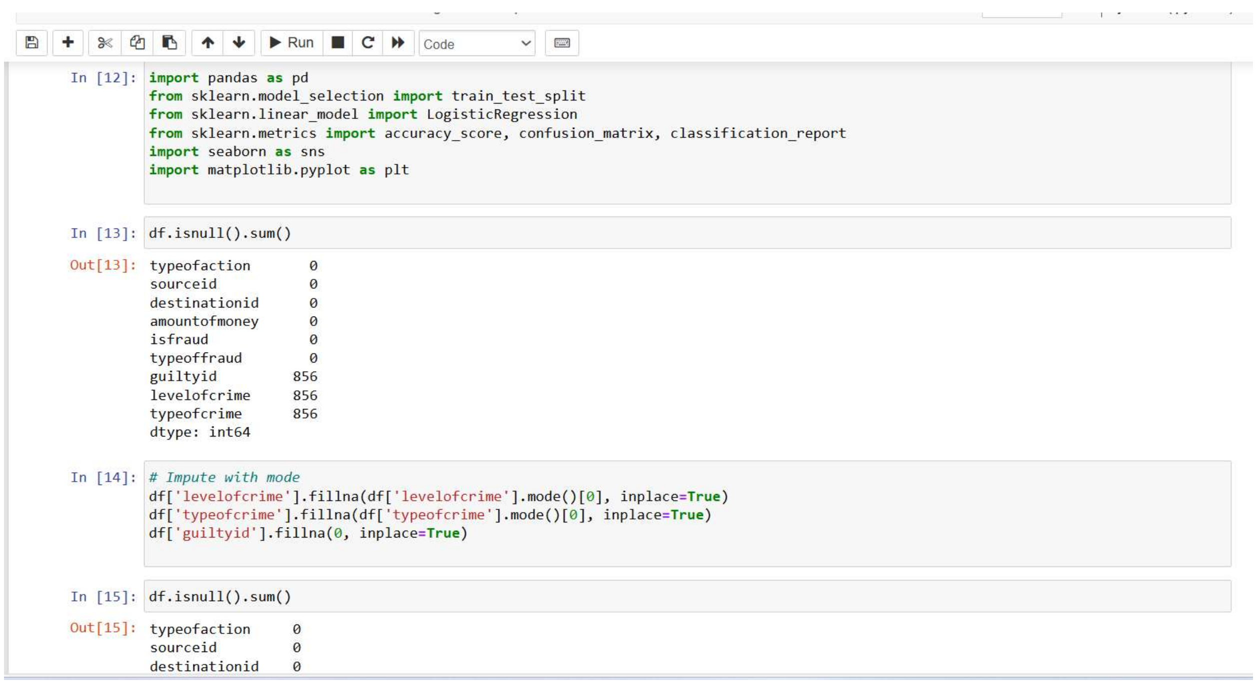


```
In [9]:  import pandas as pd
         df = pd.read_excel('ml-project 4.xlsx')

In [10]: df
Out[10]:
```

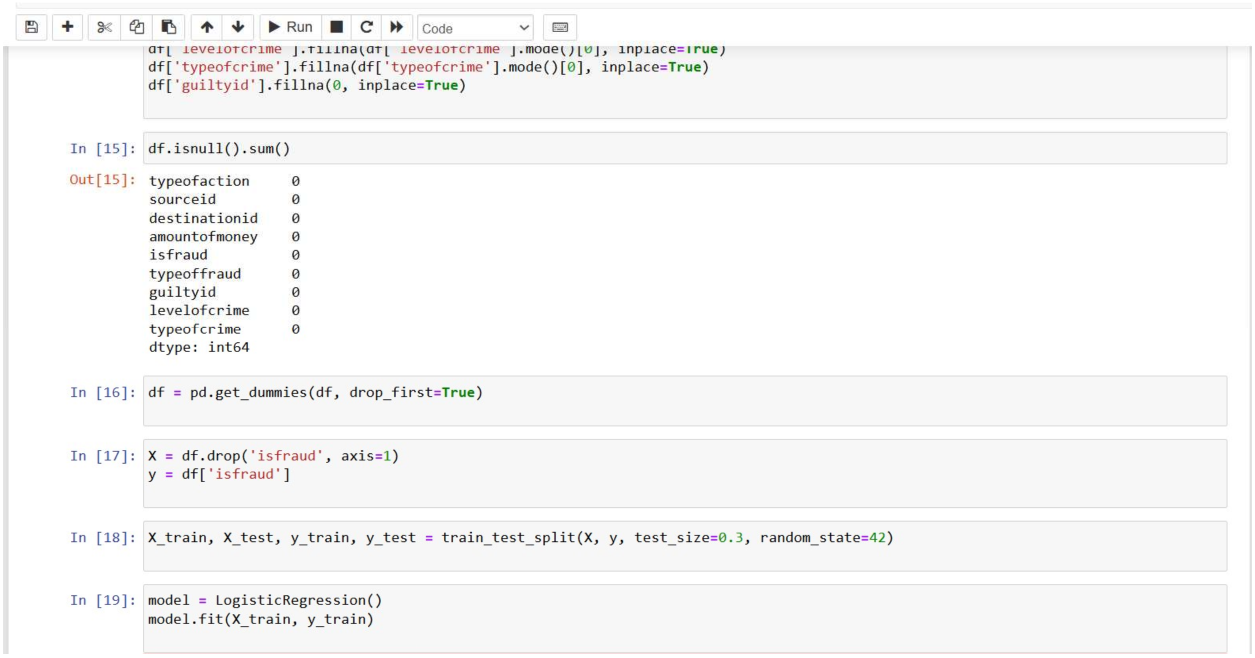
	typeofaction	sourceid	destinationid	amountofmoney	isfraud	typeoffraud	guiltyid	levelofcrime	typeofcrime
0	cash-in	30105	28942	494528	1	type1	30105.0	head	type1
1	cash-in	30105	8692	494528	1	type1	80740.0	head	type1
2	cash-in	30105	60094	494528	1	type1	92735.0	head	type1
3	cash-in	30105	20575	494528	1	type1	1615.0	head	type1
4	cash-in	30105	45938	494528	1	type1	4161.0	head	type1
...	...	...	...	...	...	...	...	...	...
2335	transfer	14945	43793	106907	0	none	NaN	NaN	NaN
2336	transfer	9532	43793	106907	0	none	NaN	NaN	NaN
2337	transfer	27332	43793	106907	0	none	NaN	NaN	NaN
2338	transfer	32685	43793	106907	0	none	NaN	NaN	NaN
2339	transfer	26390	43793	106907	0	none	NaN	NaN	NaN

2340 rows × 9 columns

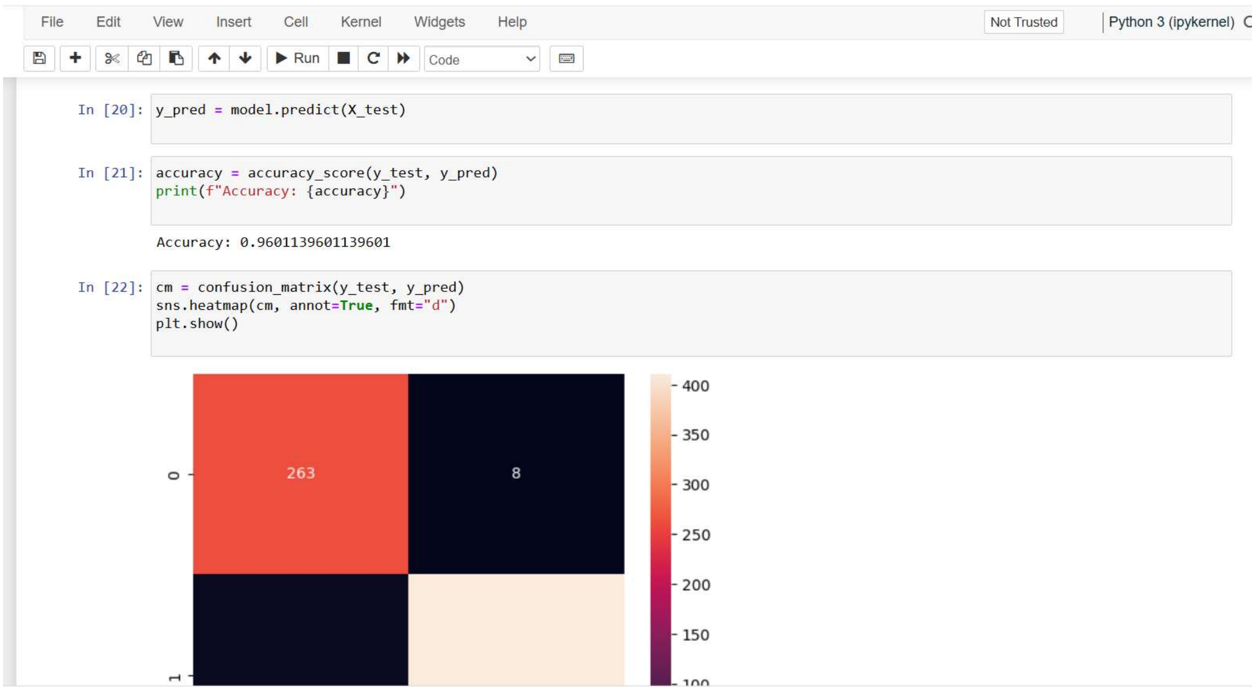
```
In [11]: df.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 2340 entries, 0 to 2339
         Data columns (total 9 columns):
```



```
In [12]: import pandas as pd
         from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
         import seaborn as sns
         import matplotlib.pyplot as plt

In [13]: df.isnull().sum()

Out[13]: typeofaction       0
         sourceid           0
         destinationid      0
         amountofmoney      0
         isfraud            0
         typeoffraud        0
         guiltyid         856
         levelofcrime     856
         typeofcrime      856
         dtype: int64

In [14]: # Impute with mode
         df['levelofcrime'].fillna(df['levelofcrime'].mode()[0], inplace=True)
         df['typeofcrime'].fillna(df['typeofcrime'].mode()[0], inplace=True)
         df['guiltyid'].fillna(0, inplace=True)

In [15]: df.isnull().sum()

Out[15]: typeofaction     0
         sourceid         0
         destinationid    0
```



```
         df['levelofcrime'].fillna(df['levelofcrime'].mode()[0], inplace=True)
         df['typeofcrime'].fillna(df['typeofcrime'].mode()[0], inplace=True)
         df['guiltyid'].fillna(0, inplace=True)

In [15]: df.isnull().sum()

Out[15]: typeofaction     0
         sourceid         0
         destinationid    0
         amountofmoney    0
         isfraud          0
         typeoffraud      0
         guiltyid         0
         levelofcrime     0
         typeofcrime      0
         dtype: int64

In [16]: df = pd.get_dummies(df, drop_first=True)

In [17]: X = df.drop('isfraud', axis=1)
         y = df['isfraud']

In [18]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

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

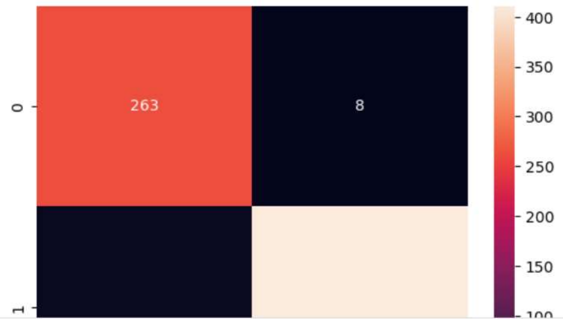


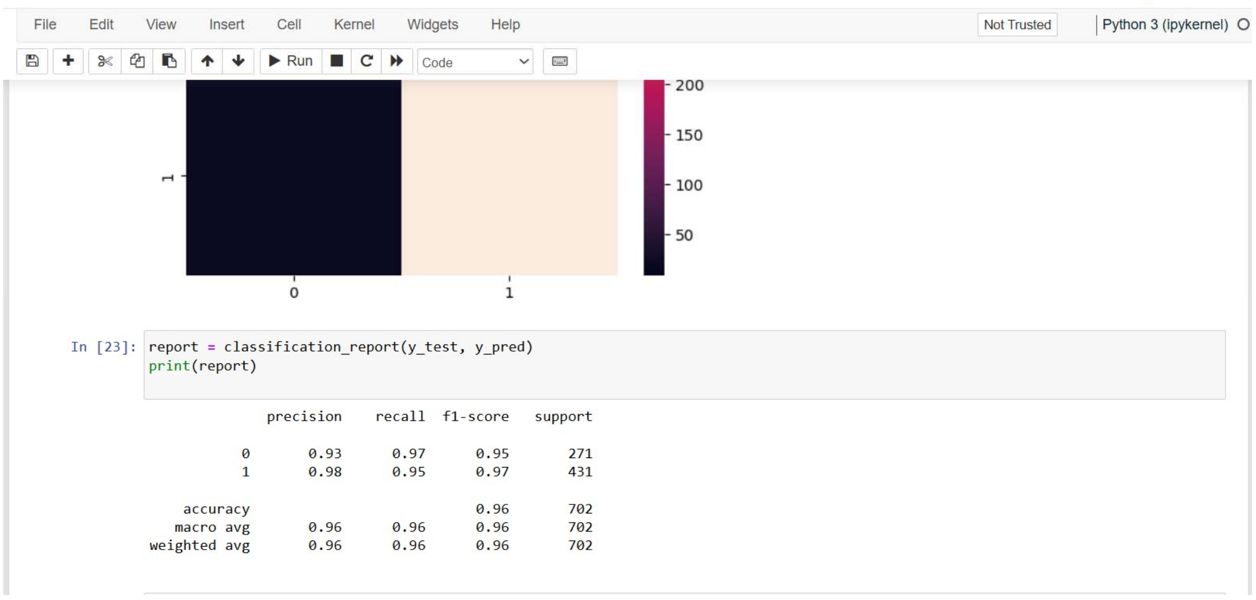
```
In [20]: y_pred = model.predict(X_test)
```

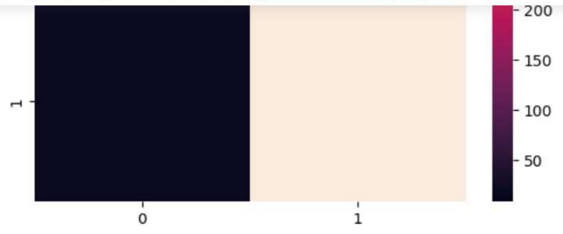
```
In [21]: accuracy = accuracy_score(y_test, y_pred)
         print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.9601139601139601
```

```
In [22]: cm = confusion_matrix(y_test, y_pred)
         sns.heatmap(cm, annot=True, fmt="d")
         plt.show()
```

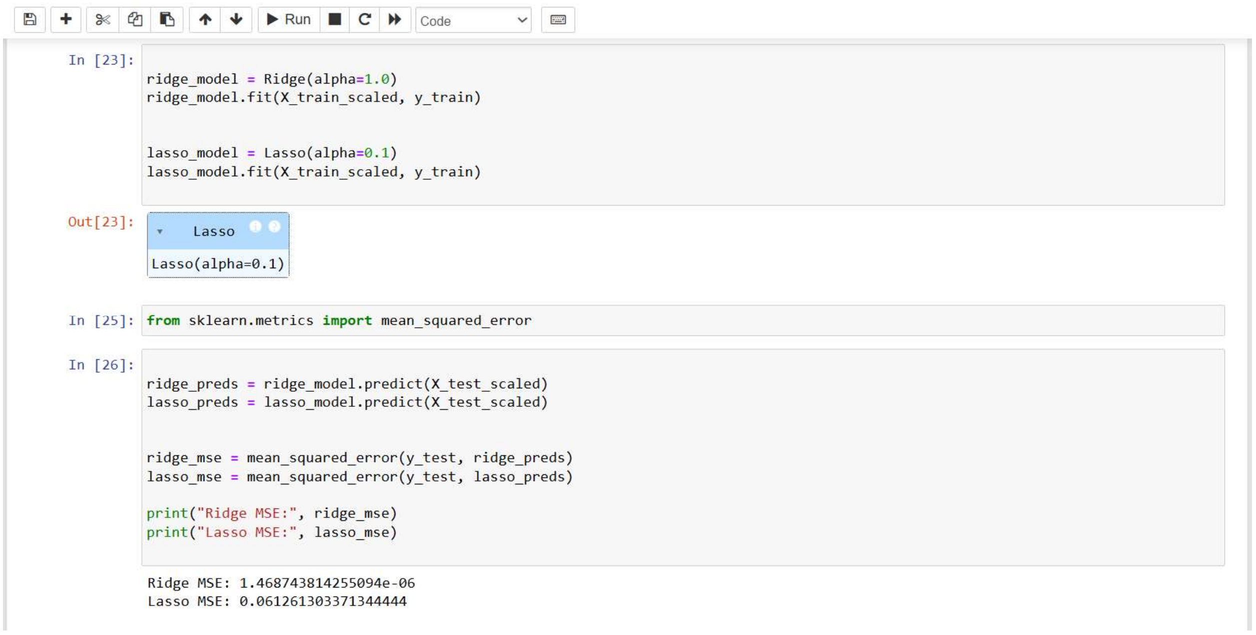






```
In [23]: report = classification_report(y_test, y_pred)
         print(report)
```

```
              precision    recall  f1-score   support

           0       0.93      0.97      0.95       271
           1       0.98      0.95      0.97       431

    accuracy                           0.96       702
   macro avg       0.96      0.96      0.96       702
weighted avg       0.96      0.96      0.96       702
```



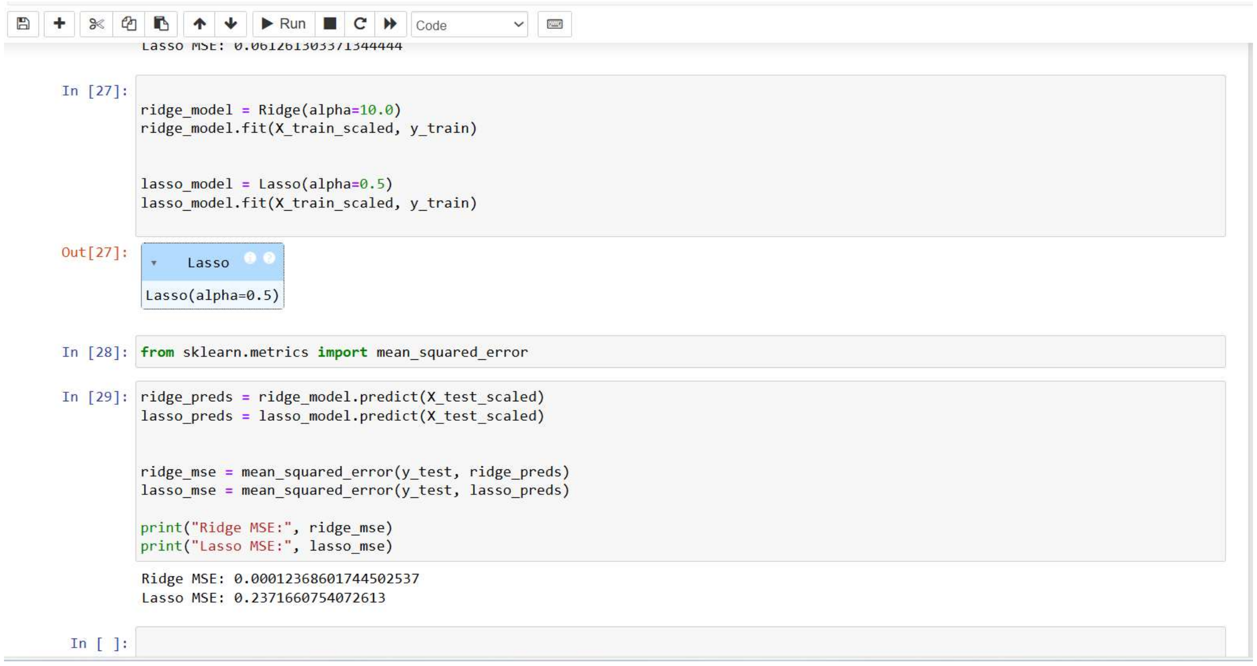
```
In [23]:
ridge_model = Ridge(alpha=1.0)
ridge_model.fit(X_train_scaled, y_train)


lasso_model = Lasso(alpha=0.1)
lasso_model.fit(X_train_scaled, y_train)
```

```
Out[23]:
Lasso
Lasso(alpha=0.1)
```

```
In [25]: from sklearn.metrics import mean_squared_error
```

```
In [26]:
ridge_preds = ridge_model.predict(X_test_scaled)
lasso_preds = lasso_model.predict(X_test_scaled)


ridge_mse = mean_squared_error(y_test, ridge_preds)
lasso_mse = mean_squared_error(y_test, lasso_preds)

print("Ridge MSE:", ridge_mse)
print("Lasso MSE:", lasso_mse)
```

```
Ridge MSE: 1.468743814255094e-06
Lasso MSE: 0.061261303371344444
```



```
Lasso MSE: 0.061261303371344444
```

```
In [27]:
ridge_model = Ridge(alpha=10.0)
ridge_model.fit(X_train_scaled, y_train)


lasso_model = Lasso(alpha=0.5)
lasso_model.fit(X_train_scaled, y_train)
```

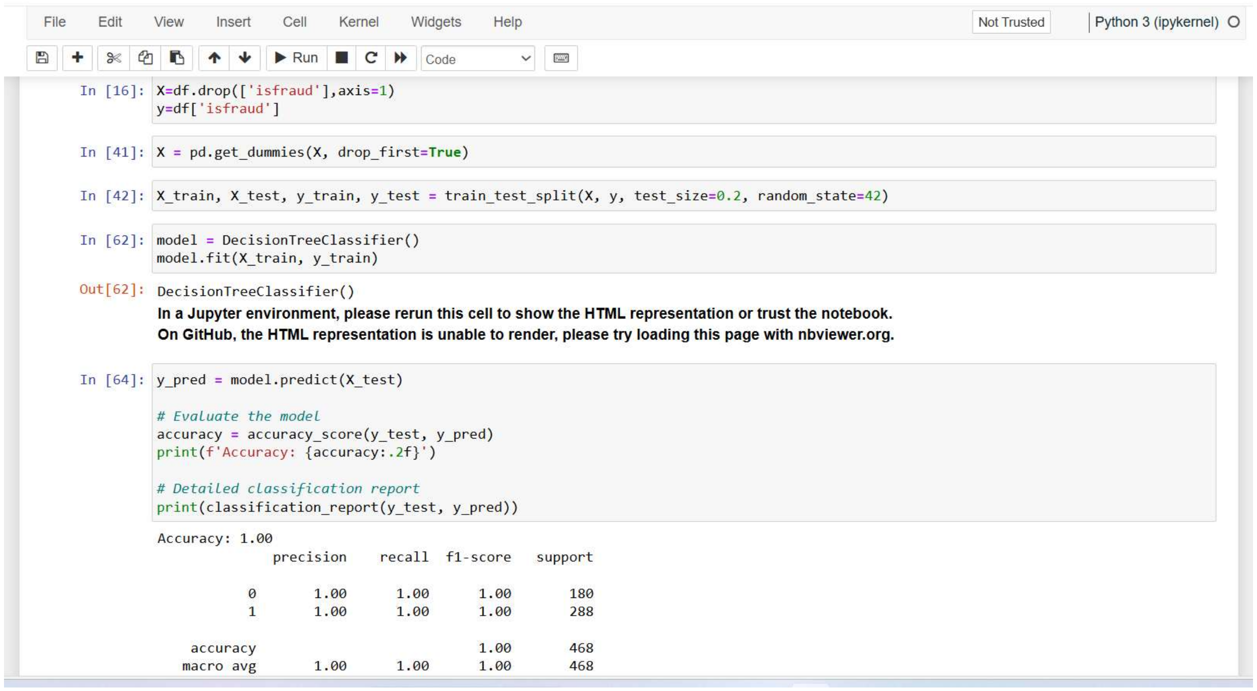
```
Out[27]:
Lasso
Lasso(alpha=0.5)
```

```
In [28]: from sklearn.metrics import mean_squared_error
```

```
In [29]: ridge_preds = ridge_model.predict(X_test_scaled)
lasso_preds = lasso_model.predict(X_test_scaled)


ridge_mse = mean_squared_error(y_test, ridge_preds)
lasso_mse = mean_squared_error(y_test, lasso_preds)

print("Ridge MSE:", ridge_mse)
print("Lasso MSE:", lasso_mse)
```

```
Ridge MSE: 0.00012368601744502537
Lasso MSE: 0.2371660754072613
```

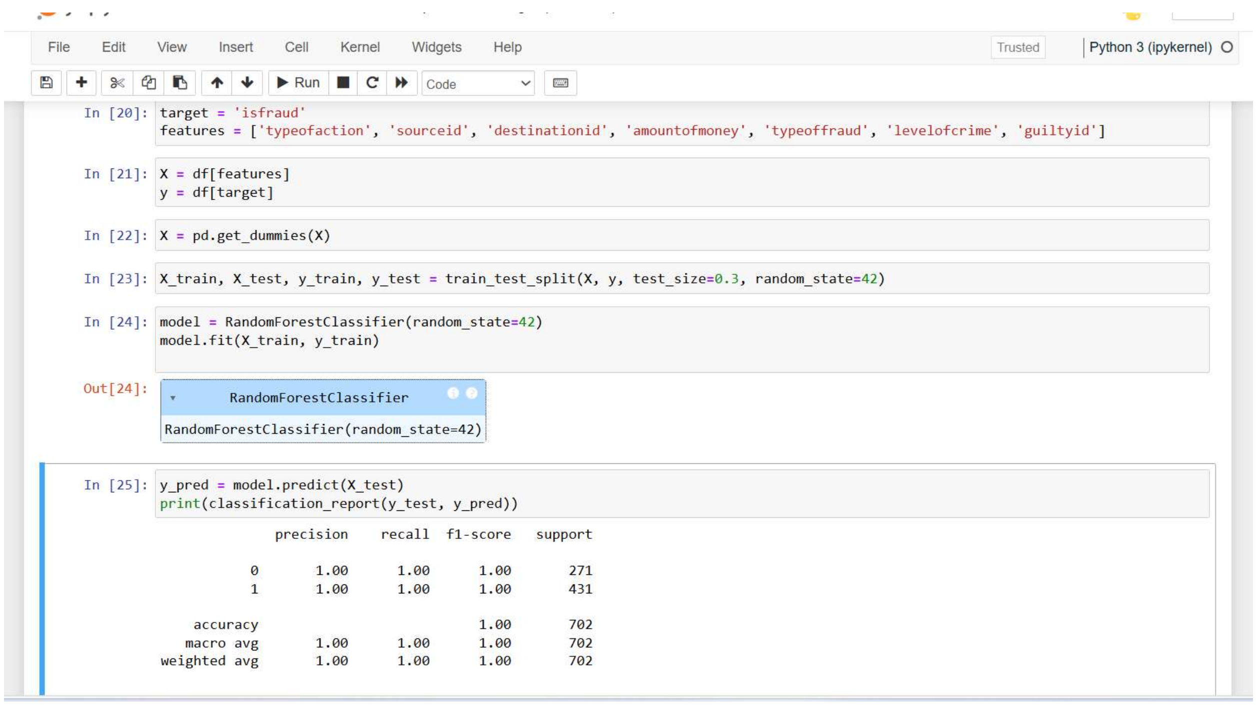
```
In [ ]:
```



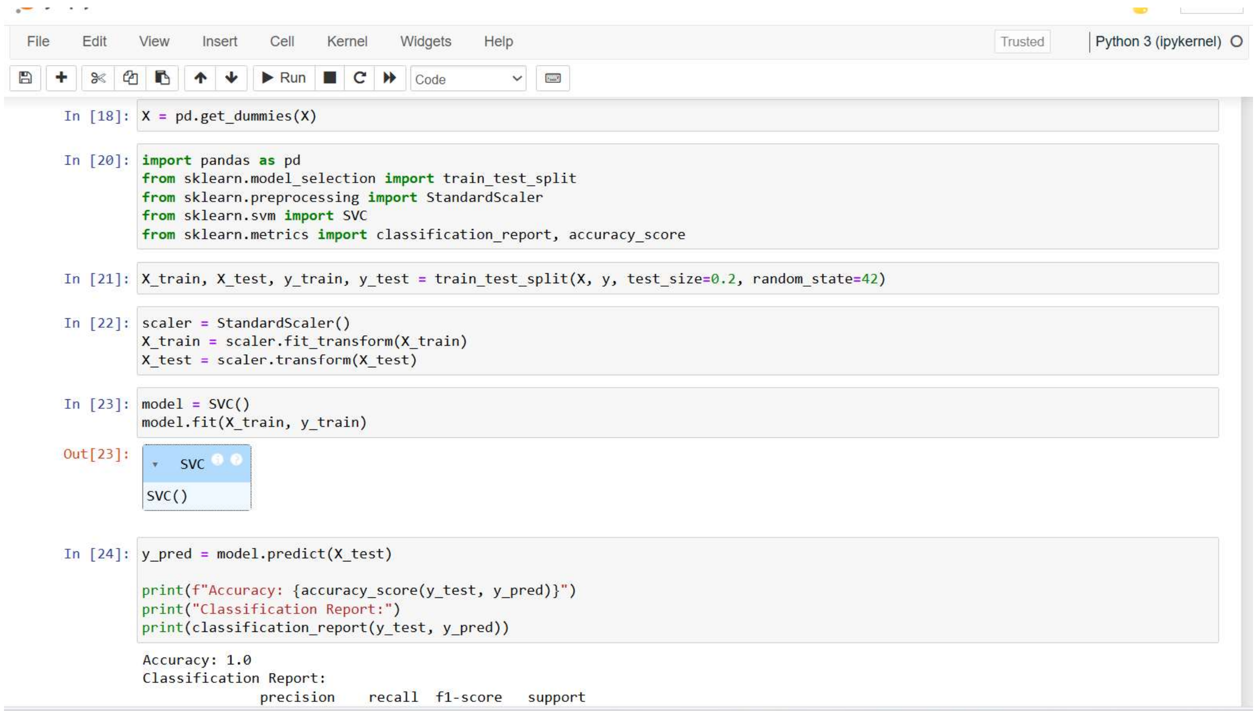
```
In [16]: X=df.drop(['isfraud'],axis=1)
         y=df['isfraud']

In [41]: X = pd.get_dummies(X, drop_first=True)

In [42]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

In [62]: model = DecisionTreeClassifier()
         model.fit(X_train, y_train)

Out[62]: DecisionTreeClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

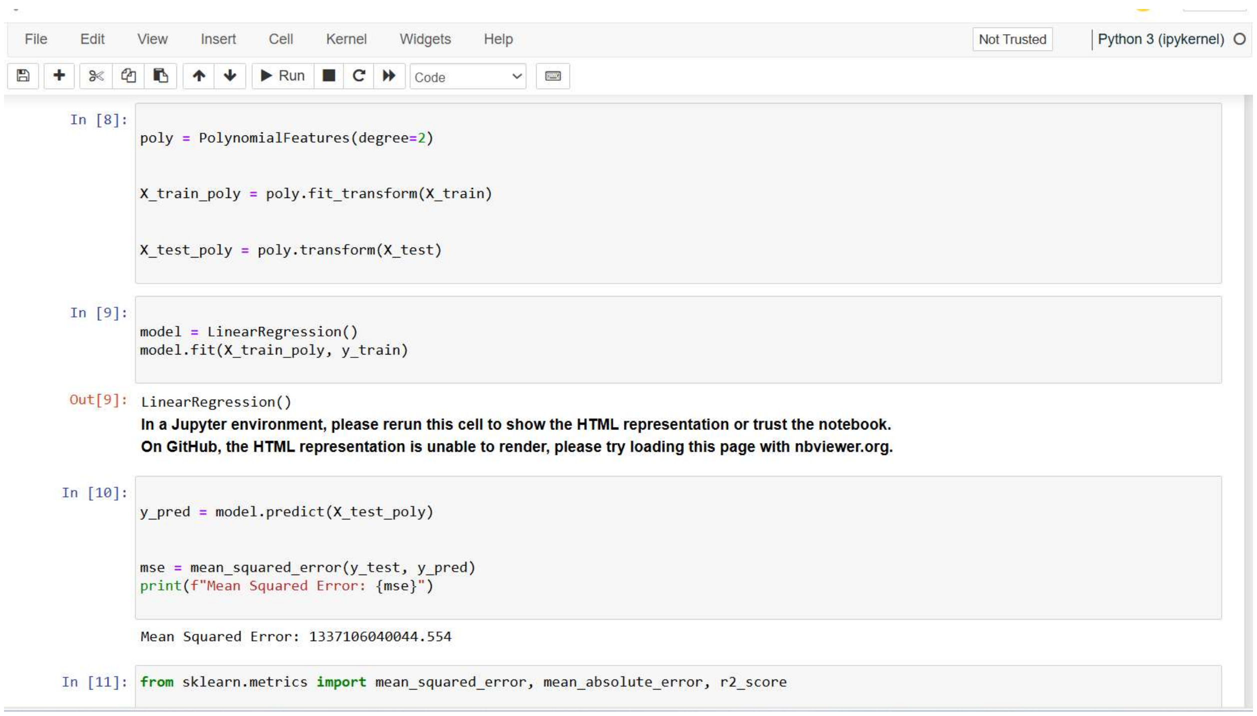
```
In [64]: y_pred = model.predict(X_test)

         # Evaluate the model
         accuracy = accuracy_score(y_test, y_pred)
         print(f'Accuracy: {accuracy:.2f}')

         # Detailed classification report
         print(classification_report(y_test, y_pred))

         Accuracy: 1.00
                       precision    recall  f1-score   support

                    0       1.00      1.00      1.00       180
                    1       1.00      1.00      1.00       288

             accuracy                           1.00       468
            macro avg       1.00      1.00      1.00       468
```



```
In [20]: target = 'isfraud'
         features = ['typeofaction', 'sourceid', 'destinationid', 'amountofmoney', 'typeoffraud', 'levelofcrime', 'guiltyid']

In [21]: X = df[features]
         y = df[target]

In [22]: X = pd.get_dummies(X)

In [23]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

In [24]: model = RandomForestClassifier(random_state=42)
         model.fit(X_train, y_train)

Out[24]: RandomForestClassifier
         RandomForestClassifier(random_state=42)

In [25]: y_pred = model.predict(X_test)
         print(classification_report(y_test, y_pred))

                       precision    recall  f1-score   support

                    0       1.00      1.00      1.00       271
                    1       1.00      1.00      1.00       431

             accuracy                           1.00       702
            macro avg       1.00      1.00      1.00       702
         weighted avg       1.00      1.00      1.00       702
```



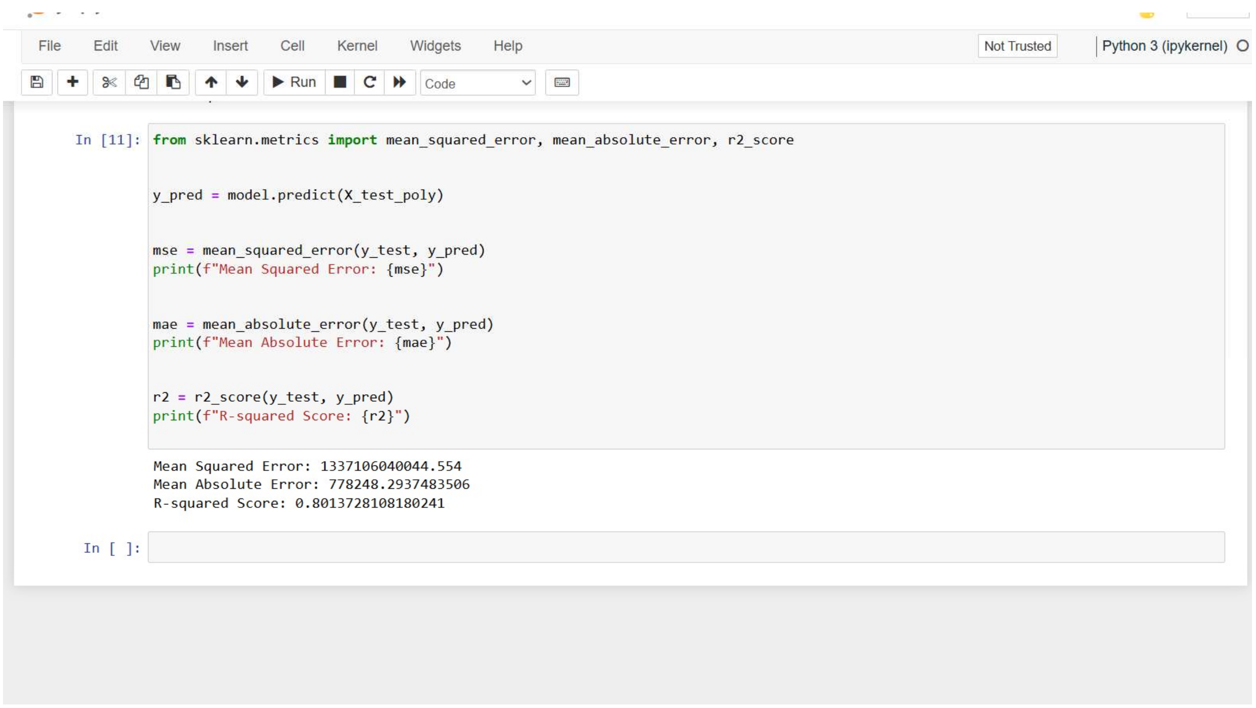
```
In [18]: X = pd.get_dummies(X)

In [20]: import pandas as pd
         from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import StandardScaler
         from sklearn.svm import SVC
         from sklearn.metrics import classification_report, accuracy_score

In [21]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

In [22]: scaler = StandardScaler()
         X_train = scaler.fit_transform(X_train)
         X_test = scaler.transform(X_test)

In [23]: model = SVC()
         model.fit(X_train, y_train)

Out[23]: SVC
         SVC()

In [24]: y_pred = model.predict(X_test)

         print(f"Accuracy: {accuracy_score(y_test, y_pred)}")
         print("Classification Report:")
         print(classification_report(y_test, y_pred))

         Accuracy: 1.0
         Classification Report:
                       precision    recall  f1-score   support
```



```
In [8]:
         poly = PolynomialFeatures(degree=2)


         X_train_poly = poly.fit_transform(X_train)


         X_test_poly = poly.transform(X_test)

In [9]:
         model = LinearRegression()
         model.fit(X_train_poly, y_train)

Out[9]: LinearRegression()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [10]:
         y_pred = model.predict(X_test_poly)


         mse = mean_squared_error(y_test, y_pred)
         print(f"Mean Squared Error: {mse}")

         Mean Squared Error: 1337106040044.554

In [11]: from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```



```
In [11]: from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score


         y_pred = model.predict(X_test_poly)


         mse = mean_squared_error(y_test, y_pred)
         print(f"Mean Squared Error: {mse}")


         mae = mean_absolute_error(y_test, y_pred)
         print(f"Mean Absolute Error: {mae}")


         r2 = r2_score(y_test, y_pred)
         print(f"R-squared Score: {r2}")
```

```
Mean Squared Error: 1337106040044.554
Mean Absolute Error: 778248.2937483506
R-squared Score: 0.8013728108180241
```

```
In [ ]:
```