TikTok Video Trends

By Eric Lin

Model Purpose

- Be able to predict a TikTok video’s view count from a variety of features.

- Find out what features may have an effect on Tiktok view count.

- Find out if there is a best model of regression for view count.

Features

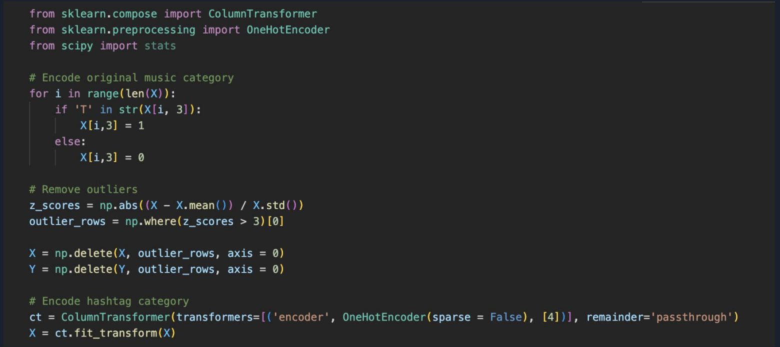
1. Creator’s follower count.
2. Creator’s following count.
3. Creator’s total likes.
4. The originality of the video sound.
5. The most popular hashtag used.

Preprocessing Phase

1. Select features and designate what values I am looking for.

2. Transform Categorical Data
   a. Replace the True/False values in the “Original Music” category to 1s and 0s.
   b. Use ColumnTransformer and OneHotCoder to encode the “Hashtag” category.

3. Remove Outliers.

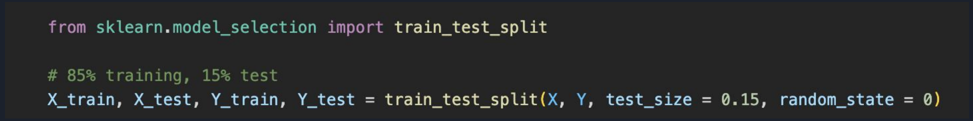


```python
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from scipy import stats

# Encode original music category
for i in range(len(X)):
    if 'T' in str(X[i, 3]):
        X[i,3] = 1
    else:
        X[i,3] = 0

# Remove outliers
z_scores = np.abs((X - X.mean()) / X.std())
outlier_rows = np.where(z_scores > 3)[0]

X = np.delete(X, outlier_rows, axis = 0)
Y = np.delete(Y, outlier_rows, axis = 0)

# Encode hashtag category
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(sparse = False), [4])], remainder='passthrough')
X = ct.fit_transform(X)
```

Splitting Data

- Training Set
  - 85%
  - Used to train the model.

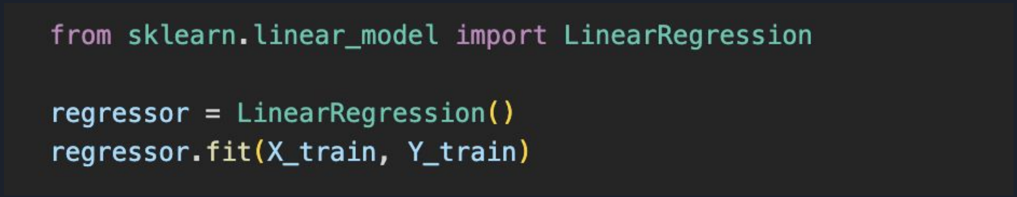
- Testing Set
  - 15%
  - Used to test against the model.



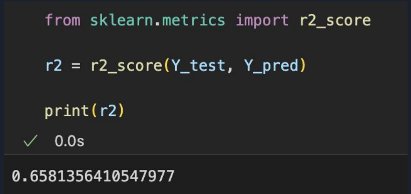
```
from sklearn.model_selection import train_test_split

# 85% training, 15% test
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.15, random_state = 0)
```

Multiple Linear Regression (MLR)

- Regression is used to determine continuous values.
- Multiple Features: 5 different features.
- Complex Relation.



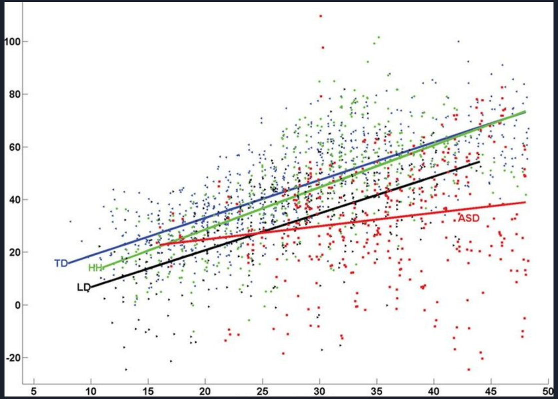
```python
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit(X_train, Y_train)
```



```python
from sklearn.metrics import r2_score

r2 = r2_score(Y_test, Y_pred)

print(r2)
```

```
✓ 0.0s
```

```
0.6581356410547977
```

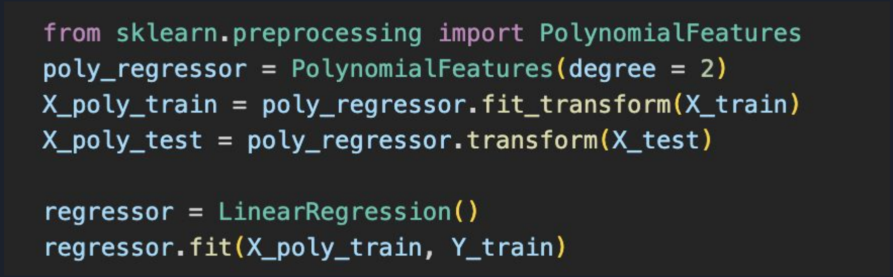


Multivariate Polynomial Regression (MPR)

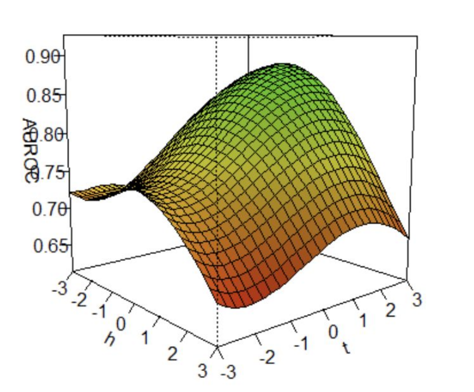
- Polynomial Regression used when there is no linear correlation.

- Multiple Features: 5 different features

- More complex than linear regression.

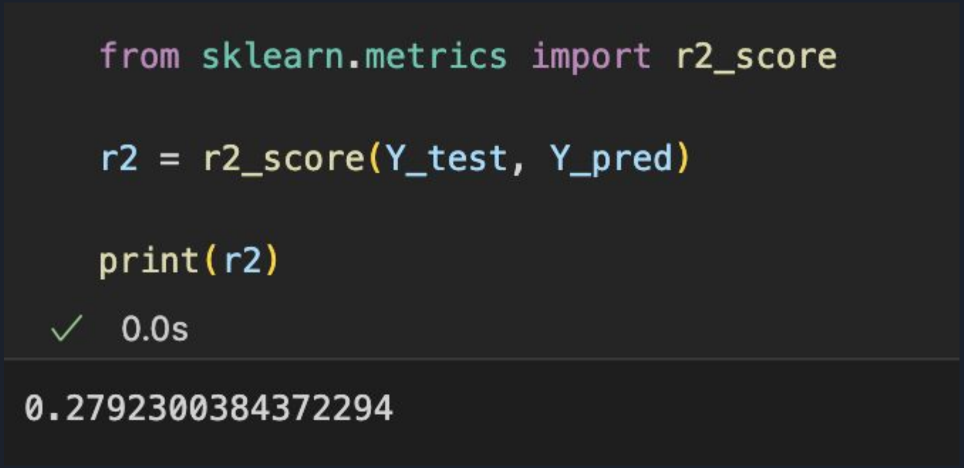


```
from sklearn.preprocessing import PolynomialFeatures
poly_regressor = PolynomialFeatures(degree = 2)
X_poly_train = poly_regressor.fit_transform(X_train)
X_poly_test = poly_regressor.transform(X_test)

regressor = LinearRegression()
regressor.fit(X_poly_train, Y_train)
```



Multivariate Polynomial Regression

- Downsides in this case.
- r-squared(MLR) > r-squared(MPR)



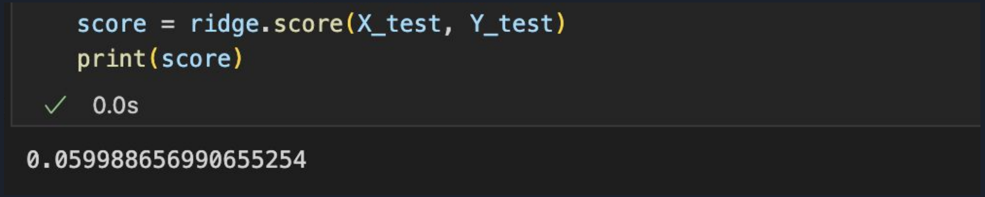
```
from sklearn.metrics import r2_score

r2 = r2_score(Y_test, Y_pred)

print(r2)
```

```
✓  0.0s
```

```
0.2792300384372294
```

Ridge Regression

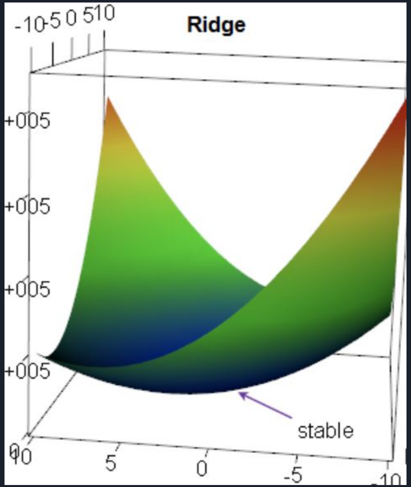
- Method of model tuning.
- Used when data suffers from multicollinearity.
- Did not work.



```
score = ridge.score(X_test, Y_test)
print(score)
```

✓ 0.0s

```
0.05998856990655254
```



Conclusion

- Multiple Linear Regression

- Fixes for the Future:
  - Larger dataset.
  - More features that are related to viewer count.