Instagram Liked Predictor

Anshul Panda

Project Overview: Predicting Instagram Likes

Objective: Classify how many Instagram likes a post will receive.

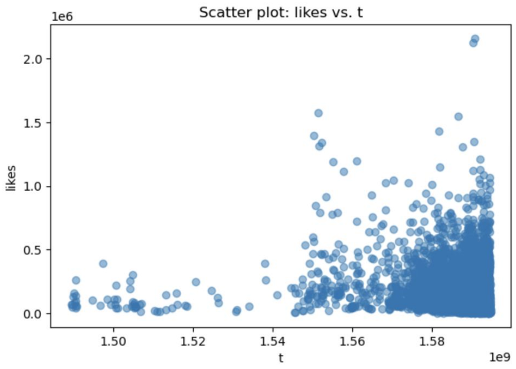
Approach: Classify posts into 3 classes (low, medium, high) based on features such as image properties and number of followers.

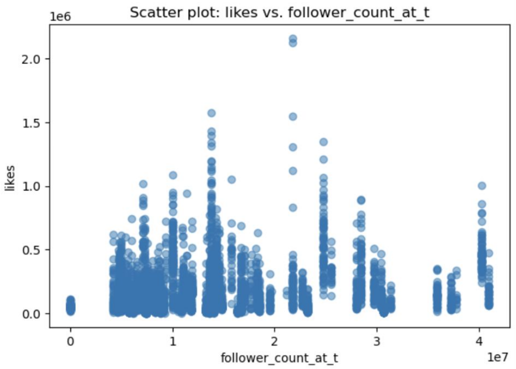
Project Overview: Predicting Instagram Likes

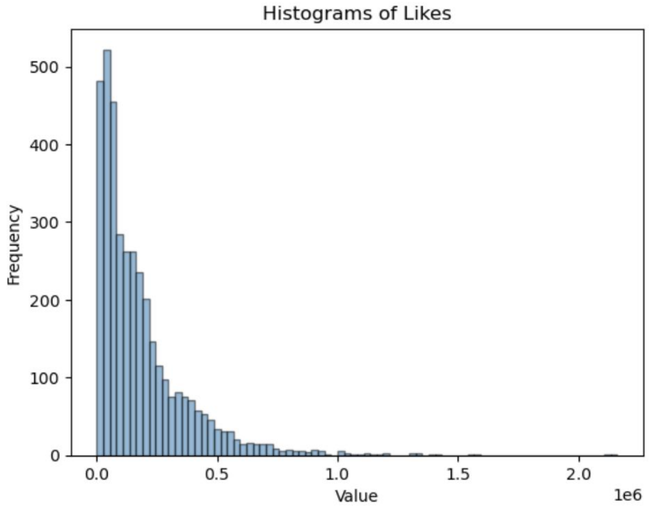
Dataset: `instagram_data.csv` with image data and follower count.

EDA Focus:

- Plotted "likes" vs. time and follower count.
- Observed clustering patterns.
- Noticed right-skewed distribution of likes.

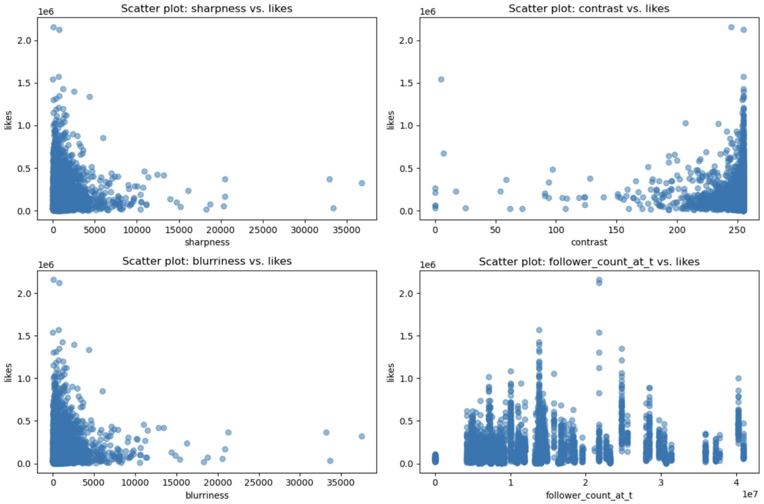






Feature Extraction

- **New Image Features**: Sharpness, contrast, and blurriness.
- **Associations**:
    - **Negative**: Sharpness and blurriness with likes.
    - **Positive**: Contrast with likes.




```python
def calculate_sharpness(image_path):
    with Image.open(image_path) as img:
        gray_img = img.convert('L')
        np_img = np.array(gray_img)
        laplacian = cv2.Laplacian(np_img, cv2.CV_64F)
        sharpness = laplacian.var()
    return sharpness

def calculate_contrast(image_path):
    with Image.open(image_path).convert('L') as img:
        np_img = np.array(img)
        min_pixel = np.min(np_img)
        max_pixel = np.max(np_img)
        contrast = max_pixel - min_pixel
    return contrast

def detect_blurriness(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    laplacian_var = cv2.Laplacian(img, cv2.CV_64F).var()
    return laplacian_var
```

```python
from tqdm import tqdm

tqdm.pandas()

instagram_data['sharpness'] = instagram_data['image_path'].progress_apply(calculate_sharpness)
instagram_data['contrast'] = instagram_data['image_path'].progress_apply(calculate_contrast)
instagram_data['blurriness'] = instagram_data['image_path'].progress_apply(detect_blurriness)
```

✓ 3m 0.0s

```
100%|██████████| 3785/3785 [01:17<00:00, 48.70it/s]
100%|██████████| 3785/3785 [00:58<00:00, 64.93it/s]
100%|██████████| 3785/3785 [00:44<00:00, 85.99it/s]
```

Clustering and Classification

- **Clustering**: Used K-means clustering to label data into 3 classes:
  - **Class 0**: 71.57% (low likes: 1431 to 218688)
  - **Class 1**: 24.02% (medium likes: 218801 to 574494)
  - **Class 2**: 4.41% (high likes: 575590 to 2161369)
- **Model Structure**:
  - Sharpness, Contrast, Blurriness -> Class (0, 1, 2)
- **Classification Models**: Tested 5 models:
  - Logistic Regression
  - KNN Classifier
  - SVM Classifier
  - Random Forest Classifier
  - Gradient Boosting Classifier

```python
features = ['sharpness', 'contrast', 'blurriness', 'follower_count_at_t']
X = instagram_data[features]
y = instagram_data['likes_class']

# 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)

# Initialize the models
models = {
    'Logistic Regression': LogisticRegression(),
    'KNN Classifier': KNeighborsClassifier(n_neighbors=3),
    'SVM Classifier': SVC(),
    'Random Forest Classifier': RandomForestClassifier(n_estimators=100, random_state=42),
    'Gradient Boosting Classifier': GradientBoostingClassifier(n_estimators=100, random_state=42)
}
```

Model Performance

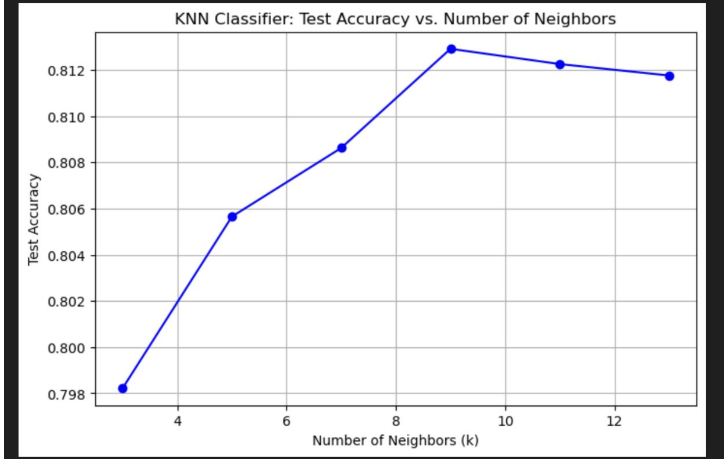
Best Models:

- Gradient Boosting: 0.7992 accuracy
- KNN: 0.7847 accuracy

Tuning: Improved KNN accuracy to 0.8129 using GridSearch (k=9).

```
Logistic Regression:
  Train Accuracy: 0.7193
  Test Accuracy: 0.7015

KNN Classifier:
  Train Accuracy: 0.8854
  Test Accuracy: 0.7847

SVM Classifier:
  Train Accuracy: 0.7193
  Test Accuracy: 0.7015

Random Forest Classifier:
  Train Accuracy: 1.0000
  Test Accuracy: 0.7596

Gradient Boosting Classifier:
  Train Accuracy: 0.8398
  Test Accuracy: 0.7992
```

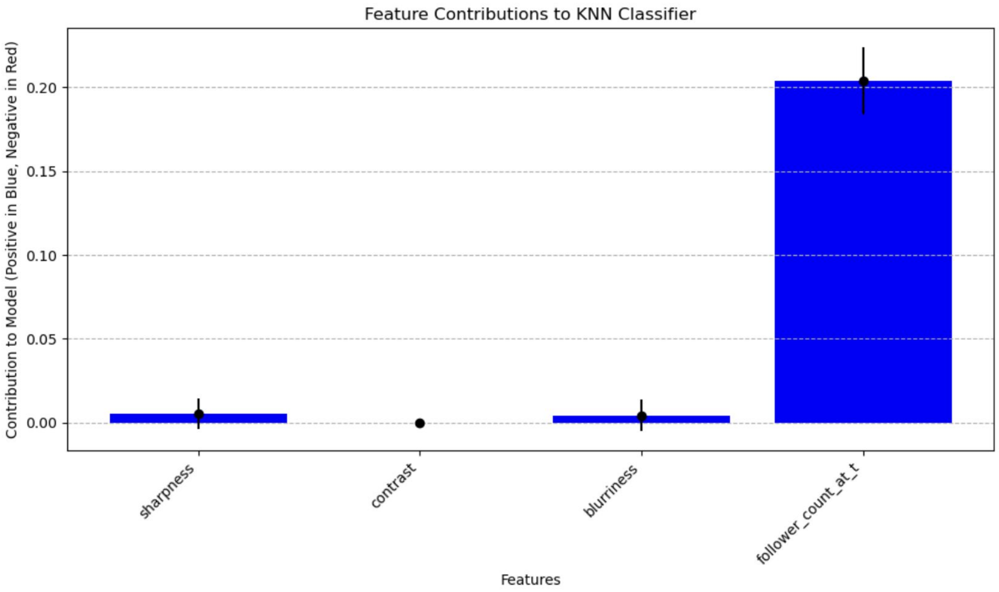
```
Optimal number of neighbors (k): 9
Test Accuracy for optimal k: 0.8129
```



KNN Feature Contributions

KNN Feature Contributions:

- **Most Important**: Follower count.
- **Secondary**: Sharpness, followed by blurriness.



Conclusions and Next Steps

- **Adaptation**: This project can be adapted to predict post success.
- **Class Creation**: Can be modified based on different success criteria.
- **Limitations**: Consider oversampling or undersampling for better balance.
- **Conclusion**: Sharpness, contrast, and blurriness are useful predictors for post success.