Python程序设计与数据科学导论期中大作业报告

徐靖 2200012917 信息科学技术学院

主题：基于观影数据集的数据分析与挖掘

一、传统偏好发现

- 衡量偏好程度的指标(下为指标函数计算代码)

In [ ]:
```
# 指标一 ：(年龄段平均打分-全局平均打分)*观影人数
score['score1'] = score['delta_rating'] * np.log(score['count'])
print(f"{value} ")

# 指标二 ：统计量：Ra=WR+(1-W)R0
alpha = 0.5
score['score2'] = alpha * score['count'] / score['count'].mean()
score['score2'] = score['score2'].apply(lambda x: min(x, 1))
score['score2'] = score['score2'] * score['rating'] + (1 - score['score2']) * me
```

- 不同年龄段用户的前10个电影(这里只展示了under 18,代码里均有输出)

Under 18
该年龄段最偏好的10部电影：(score1)

ranking	title	genres
1	Silence of the Lambs, The (1991)	Drama|Thriller
2	Lethal Weapon (1987)	Action|Comedy|Crime|Drama
3	Rear Window (1954)	Mystery|Thriller
4	Hamlet (1996)	Drama
5	Gandhi (1982)	Drama
6	Christmas Story, A (1983)	Comedy|Drama
7	Austin Powers: International Man of Mystery (1...	Comedy
8	Jaws (1975)	Action|Horror
9	Easy Rider (1969)	Adventure|Drama
10	Mystery Science Theater 3000: The Movie (1996)	Comedy|Sci-Fi

该年龄段最偏好的10部电影：(score2)

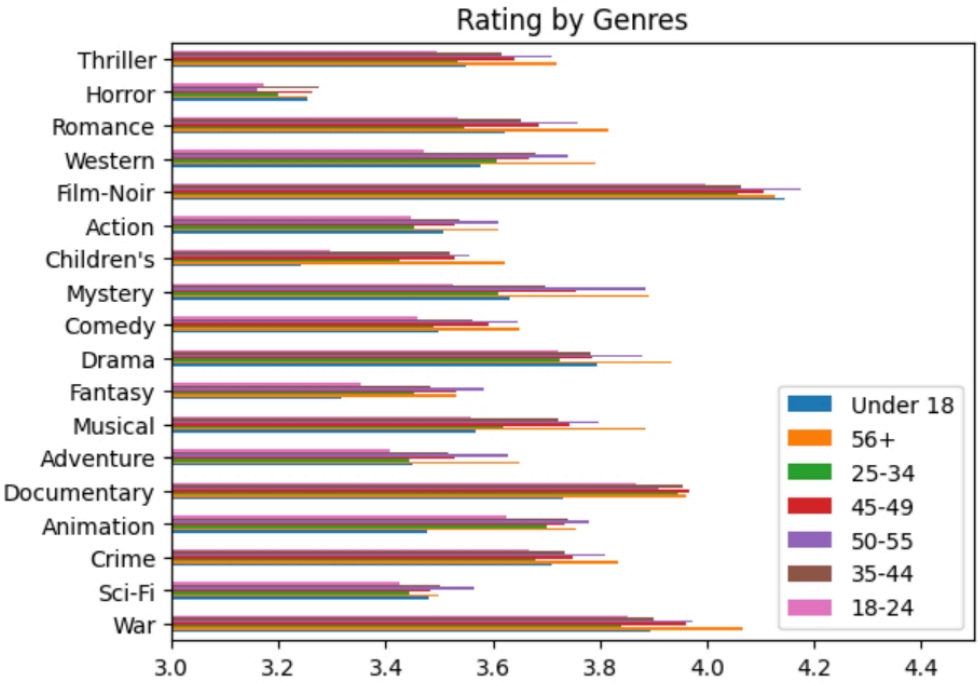
ranking	title	genres
1	Ghost Dog: The Way of the Samurai (1999)	Crime|Drama
2	Grand Day Out, A (1992)	Animation|Comedy
3	Citizen Kane (1941)	Drama
4	Midnight Cowboy (1969)	Drama
5	Bridge on the River Kwai, The (1957)	Drama|War
6	Butch Cassidy and the Sundance Kid (1969)	Action|Comedy|Western
7	Ideal Husband, An (1999)	Comedy
8	Dead Man Walking (1995)	Drama
9	Alive (1993)	Drama
10	Silence of the Lambs, The (1991)	Drama|Thriller

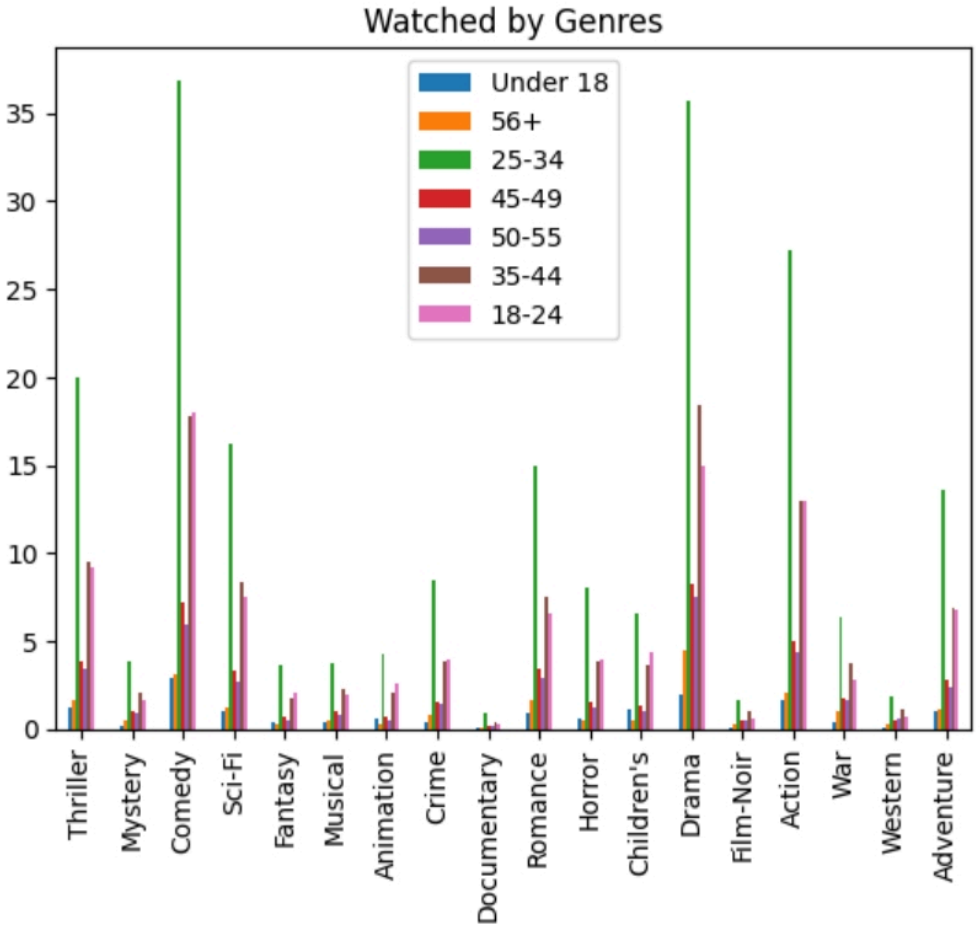
- 基于电影风格的可视化
  - 用于可视化的数据处理

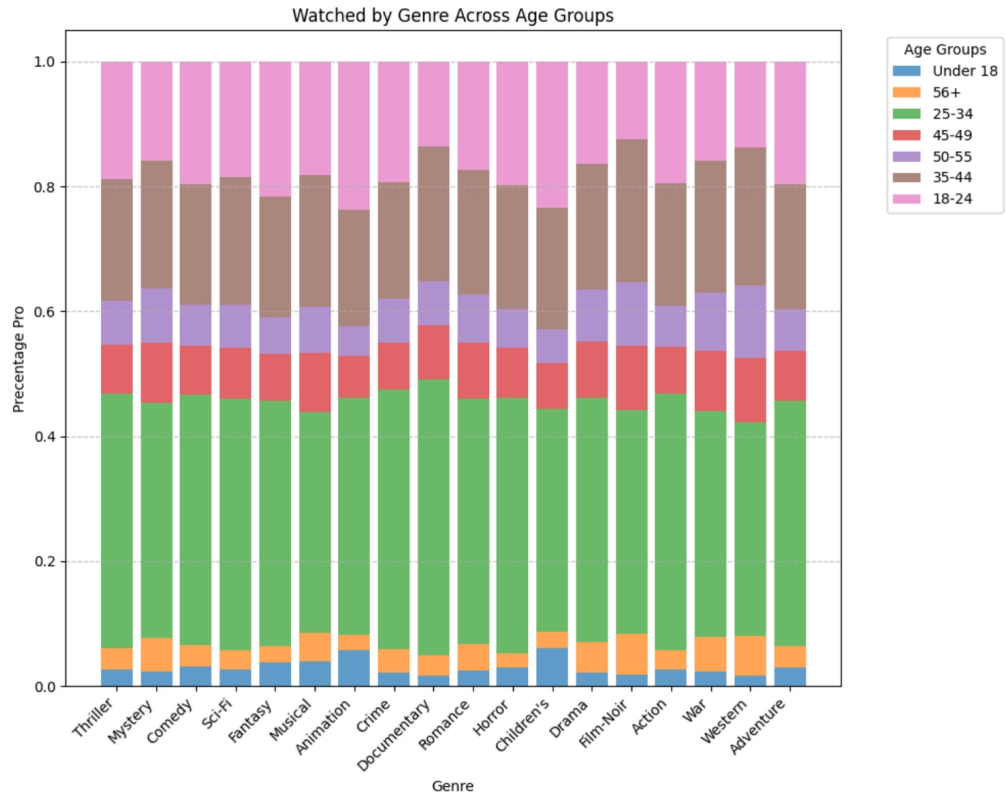
In [ ]:
```
for i in genre_list:
    for j in unique_values:
        data_2 = datas[j]
        data_2_genre = data_2[data_2[i] == 1]
        mean = data_2_genre.rating.mean()
        std = data_2_genre.rating.std()
        data_2_genre.rating = (data_2_genre.rating - mean) / std
        df_2.loc[i, (j, 'mean')] = mean
        df_2.loc[i, (j, 'std')] = std
        df_2.loc[i, (j, 'count')] = data_2_genre.shape[0]
        df_2.loc[i, (j, 'portion')] = df_2.loc[i, (j, 'count')] / num_movies
        genre_rating.append(data_2_genre.rating.to_list())

for i in genre_list:
    sumterm = 0
    for j in unique_values:
        sumterm += df_2.loc[i, (j, 'count')]
```

```
    for j in unique_values:
        df_2.loc[i, (j, 'precentage')] = df_2.loc[i, (j, 'count')] / sumterm

df_2
```

- 不同年龄用户对不同类型电影的评分
- 不同年龄用户对不同类型电影的观看数量
- 不同年龄用户占不同类型电影观看数量比率







二、用户对电影的打分预测

1.特征工程

- Onehot编码

In [ ]:
```
df = pd.concat([df, pd.get_dummies(df['age_desc'])], axis=1)
df = pd.concat([df, pd.get_dummies(df['occ_desc'])], axis=1)
df = pd.concat([df, pd.get_dummies(df['gender'])], axis=1)
df.drop(['timestamp', 'zipcode','genres','title','gender','age_desc','occ_desc']
```

- PCA降维

In [ ]:
```
pca = PCA(n_components=0.9)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
```

- 使用TF-IDF,将电影简介转化为特征

In [ ]:
```
tfidf_vectorizer = TfidfVectorizer(max_features=100)  # Choose the top 100 most
intro_tfidf_features = tfidf_vectorizer.fit_transform(df['intro'])
intro_tfidf_df = pd.DataFrame(intro_tfidf_features.toarray(), columns=tfidf_vect
df.reset_index(drop=True, inplace=True)
df = pd.concat([df, intro_tfidf_df], axis=1)
```

2.模型训练

- 数据集拆分

In [ ]:
```
y = df['rating']
X = df.drop(['rating','movie_id','user_id'] ,axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_
```

- 选用神经网络进行训练

In [ ]:
```
model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train_pca.shape[1],)),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(1, activation='linear')  # 线性激活函数用于回归任务
])

model.compile(optimizer='adam', loss='mean_squared_error')
history = model.fit(X_train_pca, y_train, epochs=10, batch_size=64, validation_s
```

3. numpy实现mse函数,并进行预测

In [ ]:
```
y_test_pred = model.predict(X_test_pca)
mse = np.mean((y_test.values-y_test_pred.transpose())**2)
print(mse)
```

- 得到的mse
  - 最后去掉了PCA降维

```
5274/5274 [==============================] - 3s 580us/step
0.9400038811918703
```

三、海报按内容聚类

1. 图像特征提取

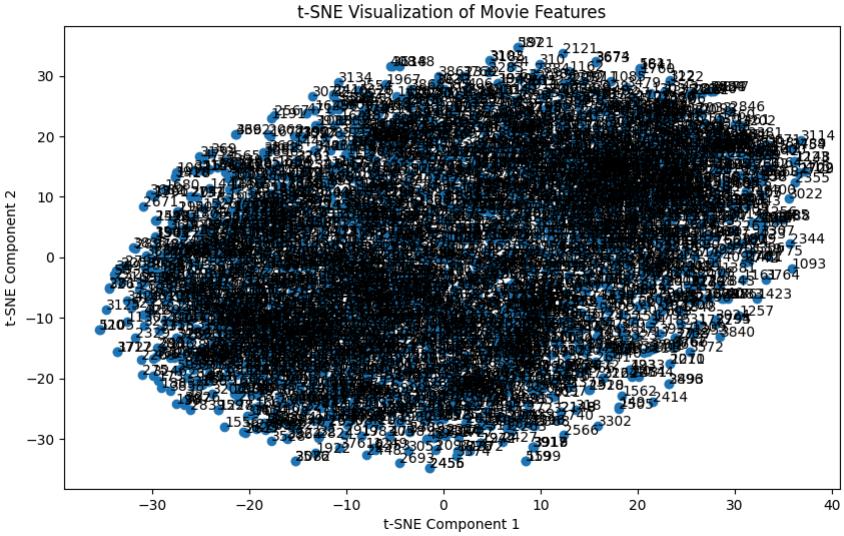
In [ ]:
```
# 提取颜色直方图和灰度直方图特征
color_hist = cv2.calcHist([image], [0, 1, 2], None, [8, 8, 8], [0, 256, 0, 256,
color_hist = cv2.normalize(color_hist, color_hist).flatten()  # 归一化并展开成一

gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
gray_hist = cv2.calcHist([gray_image], [0], None, [256], [0, 256])
gray_hist = cv2.normalize(gray_hist, gray_hist).flatten()  # 归一化并展开成一维数

# 使用Img2Vec模型提取特征向量
image_pil = Image.open(image_path)
if image_pil.mode != 'RGB':
    image_pil = image_pil.convert('RGB')
vector = img2vec_model.get_vec(image_pil)

# 拼接特征向量
# feature_vector = np.concatenate((color_hist, gray_hist, vector))
feature_vector = vector #显然用直方图效果不太好
```

2.降维

- pca降维

In [ ]:
```
pca = PCA(n_components=2)
pca_result = pca.fit_transform(features)
df_pca = df.copy()
df_pca = pd.concat([df_pca, pd.DataFrame(pca_result)], axis=1)
df_pca = df_pca.drop('features', axis=1)
```

- 尝试了t-SNE降维



In [ ]:
```
tsne = TSNE(n_components=2)  # 选择要降到的维度，这里选择2维
tsne_result = tsne.fit_transform(features)
# 将降维结果添加到DataFrame中
df_tsne = df.copy()
df_tsne['tsne_1'] = tsne_result[:, 0]
df_tsne['tsne_2'] = tsne_result[:, 1]
```

3.无监督聚类

In [ ]:
```
X = df_pca.drop(columns=['movie_id']).values
kmeans = KMeans(n_clusters=5)
kmeans.fit(X)
df['cluster'] = kmeans.labels_
```


Class 0
Class 0
Class 0
Class 0
Class 0
Class 1
Class 1
Class 1
Class 1
Class 1
Class 2
Class 2
Class 2
Class 2
Class 2
Class 3
Class 3
Class 3
Class 3
Class 3
Class 4
Class 4
Class 4
Class 4
Class 4

4.有监督聚类

- 划分数据集

In [ ]:
```
features = np.array(merged_df['features'].tolist())
cluster = np.array(merged_df['cluster'].tolist())
X = np.concatenate((features, cluster[:, np.newaxis]), axis=1)
y = np.array(merged_df['genre_vector'].tolist())
```

- 对比了k近邻,随机森林的效果,最终选择了使用神经网络做分类,细节见代码
  - 发现不降维比降维效果好

In [ ]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(X_train.shape[1],)),  # 输入层
    tf.keras.layers.Dense(256, activation=tf.nn.leaky_relu),  # 隐藏层
    tf.keras.layers.Dropout(0.3),  # Dropout层，丢弃30%的神经元
    tf.keras.layers.Dense(128, activation='relu'),  # 隐藏层
    tf.keras.layers.Dropout(0.5),  # Dropout层，丢弃50%的神经元
    tf.keras.layers.Dense(64, activation='relu'),  # 隐藏层
    tf.keras.layers.Dropout(0.5),  # Dropout层，丢弃50%的神经元
    tf.keras.layers.Dense(18, activation='sigmoid')  # 输出层，因为是多标签分类,
])

# 编译模型
model.compile(optimizer='adam',
              loss='binary_crossentropy',  # 多标签分类问题，使用交叉熵损失函数
              metrics=['accuracy'])

# 训练模型
model.fit(X_train, y_train, epochs=100, batch_size=64, validation_split=0.08)
```

- 最终的accuracy:

```
...
weighted avg       0.44      0.33      0.35       954
 samples avg       0.47      0.37      0.39       954

Accuracy: 0.20222634508348794
```

无监督聚类结果信息加入到模型中表现不佳:

- 无监督聚类算法可能对数据中的噪声或不相关信息产生过度敏感，导致聚类结果不稳定或不一致。这种不一致性可能会使得向监督学习模型传递的聚类信息不够准确或有误导性
- feature维度较大,数据集小,在此情形下无监督聚类结果相对feature对训练模型影响小,同时实践证明聚类本身降低accuracy,因此无法通过对feature聚类提高无监督聚类结果信息对模型的影响