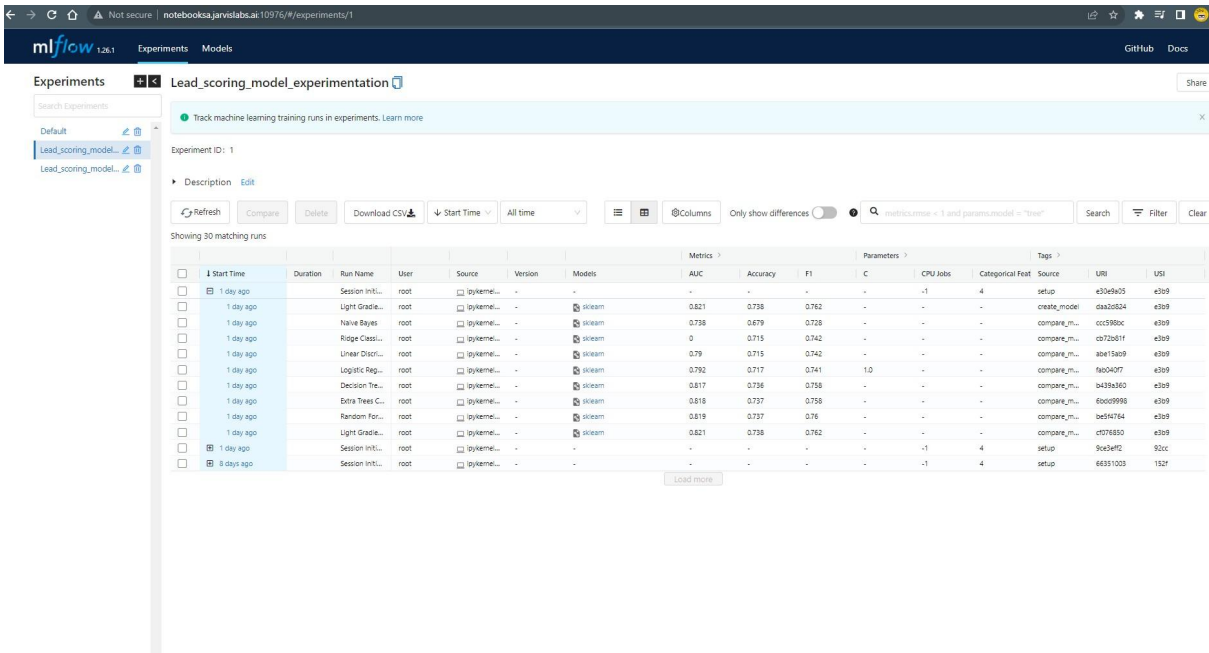
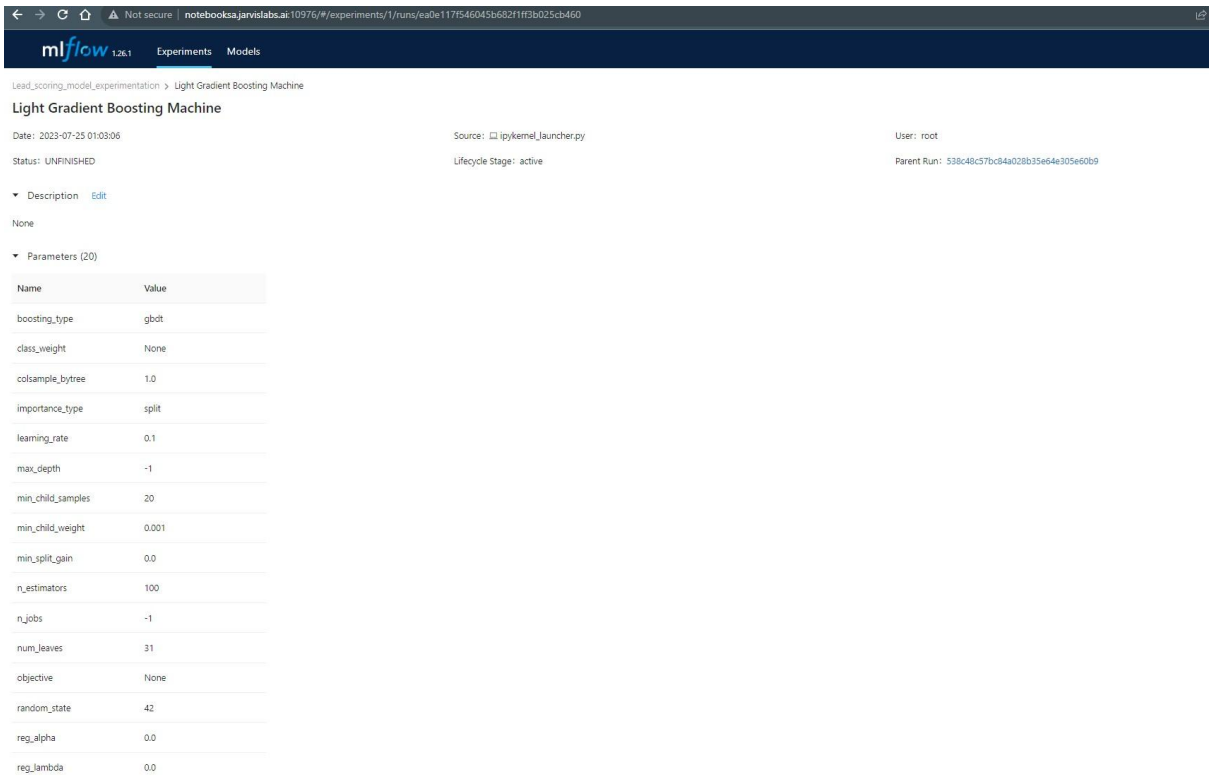


Lead_scoring_model_experimentation models



Artifacts from one of the model



Models experiments after dropping features

mlflow 1.26.1ExperimentsModelsGitHubDocs

Experiments

Lead_scoring_model_after_dropping

Search Experiments

Lead_scoring_model...
Lead_scoring_model...
Lead_scoring_model...

Experiment ID: 3

Description Edit

RefreshCompareDeleteDownload CSVStart TimeAll timeColumnsOnly show differences

Showing 12 matching runs

	Start Time	Duration	Run Name	User	Source	Version	Models	AUC	Accuracy	F1	C	CPU Jobs	Categorical Feat	Source	URI	URI
	12 minutes ago		Session Init...	root	ipykernel...	-	-	-	-	-	-	-1	\$	setup	b82c84d1	9K5
	9 minutes ago		Light Gradie...	root	ipykernel...	-	skearn	0.821	0.738	0.761	-	-	-	tune_model	04a7cc34	9K5
	11 minutes ago		Light Gradie...	root	ipykernel...	-	skearn	0.821	0.738	0.762	-	-	-	create_model	0256f866	9K5
	11 minutes ago		Naive Bayes	root	ipykernel...	-	skearn	0.794	0.673	0.725	-	-	-	compare_m...	28693c05	9K5
	11 minutes ago		Linear Discr...	root	ipykernel...	-	skearn	0.773	0.7	0.728	-	-	-	compare_m...	988e6fe1	9K5
	11 minutes ago		Ridge Classi...	root	ipykernel...	-	skearn	0	0.7	0.728	-	-	-	compare_m...	18b75495	9K5
	11 minutes ago		Logistic Reg...	root	ipykernel...	-	skearn	0.784	0.71	0.74	1.0	-	-	compare_m...	d773c520	9K5
	11 minutes ago		Decision Tre...	root	ipykernel...	-	skearn	0.817	0.736	0.758	-	-	-	compare_m...	f0492896	9K5
	11 minutes ago		Extra Trees C...	root	ipykernel...	-	skearn	0.816	0.736	0.758	-	-	-	compare_m...	38177782	9K5
	11 minutes ago		Random For...	root	ipykernel...	-	skearn	0.819	0.737	0.759	-	-	-	compare_m...	0217a936	9K5
	11 minutes ago		Extreme Gra...	root	ipykernel...	-	skearn	0.821	0.738	0.761	-	-	-	compare_m...	65f64026	9K5
	11 minutes ago		Light Gradie...	root	ipykernel...	-	skearn	0.821	0.738	0.762	-	-	-	compare_m...	40c1a726	9K5

Load more

Artifacts from one of the model after dropping features

mlflow 1.26.1ExperimentsModels

Lead_scoring_model_after_dropping > Light Gradient Boosting Machine

Light Gradient Boosting Machine

Date: 2023-07-26 17:29:52Source: ipykernel_launcher.pyUser: root

Status: UNFINISHEDLifecycle Stage: activeParent Run: 8035470076554f7ba4ad53cc8468ee2a

Description Edit

None

Parameters (20)

Name	Value
boosting_type	gbot
class_weight	None
colsample_bytree	1.0
importance_type	split
learning_rate	0.1
max_depth	-1
min_child_samples	20
min_child_weight	0.001
min_split_gain	0.0
n_estimators	100
n_jobs	-1
num_leaves	31
objective	None
random_state	42
reg_alpha	0.0
reg_lambda	0.0

Airflow dags

The screenshot shows the Airflow web interface. At the top, there are navigation links: Airflow, DAGs, Security, Browse, Admin, and Docs. Below the navigation bar, there are two yellow warning banners. The main section is titled 'DAGs' and contains a table of DAGs. The table has columns for DAG, Owner, Runs, Schedule, Last Run, Next Run, Recent Tasks, Actions, and Links. The DAGs listed include 'Lead_Scoring_Data_Engineering_Pipeline', 'Lead_scoring_inference_pipeline', 'Lead_scoring_mlflow_production', and several example DAGs like 'example_branch_operator', 'example_branch_datetime_operator', etc. The 'Lead_scoring_mlflow_production' DAG is highlighted.

Final model in production

The screenshot shows the mlflow web interface. At the top, there are navigation links: mlflow, Experiments, and Models. Below the navigation bar, there is a section titled 'Experiments' with a search bar and a list of experiments. The 'Lead_scoring_mlflow_production' experiment is selected. Below the experiment name, there is a description and a table of runs. The table has columns for Start Time, Duration, Run Name, User, Source, Version, Models, Metrics, and Parameters. The table shows one run with a duration of 42s and a model of 'LightGBM/v1'.

Final model in prod 2

The screenshot shows the mlflow web interface. At the top, there are navigation links: mlflow, Experiments, and Models. Below the navigation bar, there is a section titled 'Registered Models'. Below this section, there is a table of models. The table has columns for Name, Latest Version, Status, Production, Last Modified, and Tags. The table shows one model named 'LightGBM' with a latest version of 'Version 1' and a status of 'Production'.

The screenshot shows the mlflow web interface. At the top, there's a navigation bar with the mlflow logo, version 1.26.1, and tabs for Experiments and Models. Below the navigation bar, the breadcrumb path is "Lead_scoring_mlflow_production > run_LightGB". The main header for the selected experiment is "run_LightGB".

Metadata information is displayed:

- Date: 2023-07-26 18:58:18
- Duration: 4.2s
- Source: airflow
- User: root
- Status: FINISHED
- Lifecycle Stage: active

There are two expandable sections:

- Description**: Currently shows "None". There is an "Edit" link next to it.
- Parameters (20)**: Expanded to show a table of parameters.

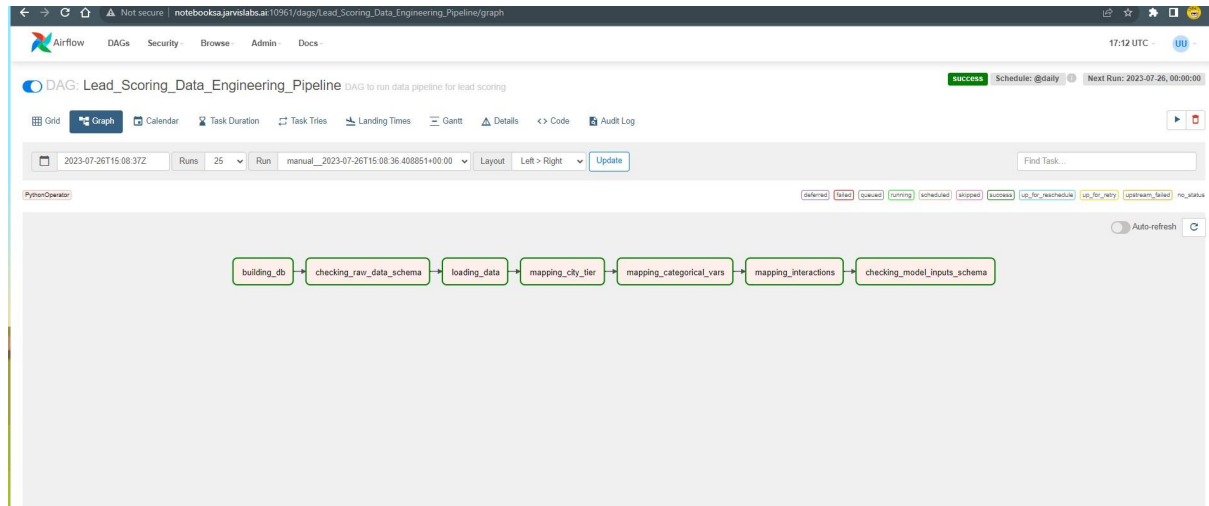
Name	Value
boosting_type	gbdt
class_weight	None
colsample_bytree	1.0
importance_type	split
learning_rate	0.1
max_depth	-1
min_child_samples	20
min_child_weight	0.001
min_split_gain	0.0
n_estimators	100
n_jobs	-1
num_leaves	31
objective	None
random_state	42
reg_alpha	0.0
reg_lambda	0.0

At the bottom left, the URL "notebooks.jarvislabs.ai/10960/#/" is visible.

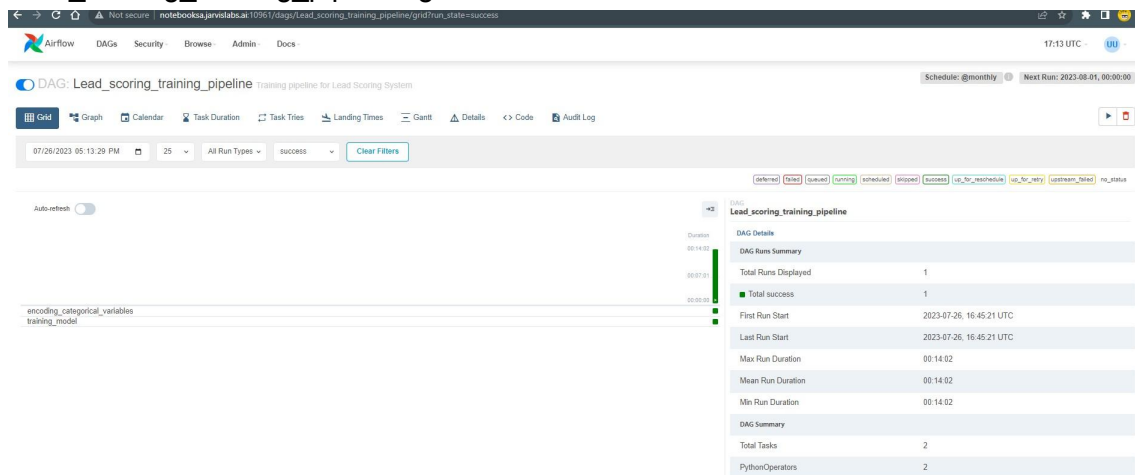
The screenshot displays the Airflow web interface for the DAG 'Lead_Scoring_Data_Engineering_Pipeline'. The top navigation bar includes links for Grid, Graph, Calendar, Task Duration, Task Tries, Landing Times, Gantt, Details, and Audit Log. The DAG is scheduled to run daily at 00:00:00 on 2023-07-26. The left sidebar lists the tasks: building_db, checking_raw_data_schema, loading_data, mapping_city_tier, mapping_categorical_vars, mapping_interactions, and checking_model_inputs_schema. The right sidebar shows the DAG run summary, including the total number of runs displayed (4), the status of the runs (3 successful, 1 failed), and the first and last run start times. The central area features a bar chart showing the duration of the runs, with a table below it providing the exact duration for each run.

Run ID	Duration
1	00:02:03
2	00:01:28
3	00:01:00
4	00:01:00
5	00:01:00
6	00:01:00
7	00:01:00
8	00:01:00
9	00:01:00
10	00:01:00
11	00:01:00
12	00:01:00
13	00:01:00
14	00:01:00
15	00:01:00
16	00:01:00
17	00:01:00
18	00:01:00
19	00:01:00
20	00:01:00

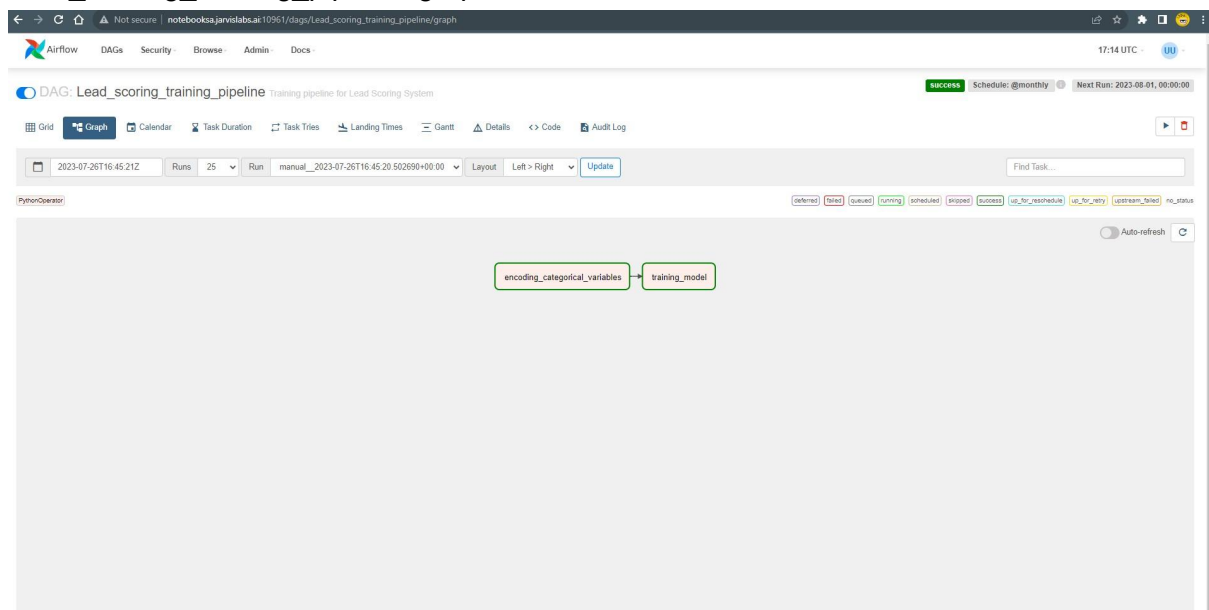
Lead_scoring_data_pipeline graph view



Lead_scoring_training_pipeline grid view



Lead_scoring_training_pipeline graph view



Not secure | notebooks.jarvislab.ai:10951/dags/Lead_scoring_inference_pipeline/grid/run_state=success

17:15 UTC

DAG: Lead_scoring_inference_pipeline inference pipeline of Lead Scoring system

Schedule: @hourly Next Run: 2023-07-26, 17:00:00

Grid Graph Calendar Task Duration Task Times Landing Times Gantt Details <> Code Audit Log

07/26/2023 05:14:52 PM 25 All Run Types success Clear Filters

Auto-refresh

encoding_categorical_variables
checking_input_features
generating_models_prediction
checking_model_prediction_ratio

Lead_scoring_inference_pipeline

DAG Details

DAG Runs Summary

Run ID	Run Status	Start Time	End Time
1	success	2023-07-26, 17:07:56 UTC	2023-07-26, 17:07:56 UTC

Max Run Duration: 00:00:33
Mean Run Duration: 00:00:33
Min Run Duration: 00:00:33

DAG Summary

Metric	Value
Total Tasks	4
PythonOperators	4

```
File Edit View Run Help Settings Help
~/ -- /app /land_useing_inference_pipeline/
Name Last Modified
--
constants 81 hour ago
land_useing_inference_pipeline 3 hour ago
predictor_distribution 19 hours ago

75 which we created previously.
76 It also replaces any null values present in 'total_land_proper' and
77 'refered_land' columns with 0.
78
79 INPUTS
80 DB_FILE_NAME : Name of the database file
81 DB_PATH : path where the db file should be
82 DB_DIRECTORY : path of the directory where 'land_useing.csv'
83               file is present.
84
85 OUTPUT
86 Saves the processed dataframe in the db in a table named 'landed_data'.
87 If the table with the same name already exists then the function
88 replaces it.
89
90 SUPPLY USAGE
91 ...
92 land_data_info_db()
93
94 conn = sqllite.connect(DB_PATH+DB_FILE_NAME)
95
96 #f1 = pd.read_csv(DB_DIRECTORY+'landuse_inference.csv')
97
98 #f1['total_land_proper'] = #f1['total_land_proper'].fillna(0)
99 #f1['refered_land'] = #f1['refered_land'].fillna(0)
100
101 #f1.to_sql(name='landed_data', con=conn, if_exists='replace', index=False)
102
103 conn.close()
104
105
106
107 #####
108 # Define function to map cities to their respective tier
109 #####
110
111 def map_city_tier():
112
113     This function maps all the cities to their respective tier as per the
114     mappings provided in the city_tier_mapping.csv file. If a
115     particular city's tier isn't mapped/foreseen in the city_tier_mapping.csv
116     file then the function maps that particular city to its own 'unmapped'
117     tier=3.
118
119     INPUTS
120     DB_FILE_NAME : Name of the database file
121     DB_PATH : path where the db file should be
122     city_tier_mapping : a dictionary that maps the cities to their tier
123
124     OUTPUT
125     Saves the processed dataframe in the db in a table named
```